Kai Jokinen

Teollisuusrobotin häiriöiden ja tapahtumien tallennus

Opinnäytetyö

Kevät 2015

Tekniikan yksikkö

Tietotekniikan koulutusohjelma

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Sulautetut järjestelmät

Tekijä: Kai Jokinen

Työn nimi: Teollisuusrobotin häiriöiden ja tapahtumien tallennus

Ohjaaja: Seppo Stenberg

Vuosi: 2015 Sivumäärä: 54 Liitteiden lukumäärä: 7

Tämän opinnäytetyön pääaiheena oli lukea ja tallentaa ABB-teollisuusrobotin huoltosarjaportista saatavia häiriö- ja tapahtumatietoja. Lukeminen tapahtui RS232-sarjaliikennettä hyväksikäyttäen. Luetut tiedot muokattiin ja tallennettiin Raspberry Pi -tietokoneen tiedostoon, josta ne ovat siirrettävissä myös tulevaisuudessa hankittavaan tietokantaan. Lisätoimintointoina on kaksi kappaletta käyntiaika- ja kappalelaskuria ohjelmoituna laitteen GPIO-tuloihin, nämä tiedot myös tallennettiin tiedostoon myöhempää tietokantaan siirtoa varten.

Lisäksi on suunniteltu toimintoja laitteen ylläpidettävyyteen ja suojaamiseen. Toimintoina olivat ohjelmoidun C++-kielisen ohjelman automaattinen päivittäminen sekä Raspberry Pi -laitteen tiedostojärjestelmän suojaaminen sähkökatkojen varalle rakennetulla UPS-laitteella.

Opinnäytetyössä käydään lyhyesti läpi käytetyt laitteet, lisäohjelmat, asennus- ja asetusohjeineen. Lisäksi selvitettiin erilaiset alkumäärittelyt ja toiminta uusien laitteiden perustamiselle. Ajallisesti suurimman huomion opinnäytetyössä kuitenkin on saanut C++-ohjelmointikielellä laadittu ohjelma eri toimintoineen sekä erilaisten ilmenneiden ongelmien ratkaisu ja huomioon otettavien asioiden pohdiskelu. Tämän opinnäytetyön voi ajatella olevan myös suuntaa antava opas tämänkaltaisen Raspberry Pillä toteutettavan sulautetun järjestelmän suunnittelussa ja perustamisessa.

Avainsanat: sulautetut järjestelmät, Raspberry Pi, sarjaportti, muistikortit, UPS, data loggeri, robotit

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Embedded Systems

Author: Kai Jokinen

Title of thesis: Saving industrial robot fault and event messages

Supervisor: Seppo Stenberg

Year: 2015 Number of pages: 54 Number of appendices: 7

The main topic of this thesis was to read and save error and event messages, which can be read from ABB industrial robot service serial port. The reading was done using the RS232 serial communication. The read data was modified and saved in a file on a Raspberry Pi computer. Additional features are two runtime counters and two piece counters, which are programmed to device GPIO-inputs. The data of counters is also saved on Raspberry Pi file. Both the saved files will be available for the database, which is going to be added to the system in the future.

There were also designed additional functions to protect the device's functionality and maintainability. The functions were automatically updating the C++-program and a UPS-device to protect the device file system against power failure.

The thesis briefly introduces also devices and applications, as well as installation and setting instructions. Also used initial settings and the necessary actions before the establishment of the new Raspberry Pi -device were explained. Most attention has been paid to C++-programming language, solving various problems and issues that need to be taken into account. This thesis can be seen as a guide for designing and creating this kind of embedded system with Raspberry Pi.

Keywords: embedded system, Raspberry Pi, serial port, memory cards, UPS, data logger, robots

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

C++	Käännettävä ohjelmointikieli.
Console-port	Ohjainyksikön huoltosarjaportti, jonne laite kirjoittaa mm. virhe- ja käynninaikaisia tapahtumia.
Controller	Ohjainyksikkö, joka ohjaa ja valvoo robotin toimintoja.
Cron	Linux-järjestelmiin sisältyvä ajastuspalvelu määriteltyjen tehtävien suorittamiseksi ajastetusti.
FlexPendant	ABB IRC 5 -käyttäjäpaneeli eli käsiohjain robotin opettamiseen, ohjaamiseen ja tietojen katseluun.
ISO 8373	Robotit ja robottilaitteiden ISO-standardi.
Konenäkö	Koneellinen vastine ihmisen näköaistille.
Raspberry Pi	ARM-suorittimeen perustuva pienikokoinen ja edullinen tietokone.
Raspbian	Ilmainen Raspberry Pille optimoitu käyttöjärjestelmä, joka pohjautuu Debian Linuxiin.
Scripti	Linux-järjestelmissä ohjelmoitava komentosarjatiedosto, joka suorittaa ohjelmoituja toimintoja yhdellä käskyllä.
Tietokanta	Digitaalisessa muodossa oleva loogisesti yhteenkuuluvien tietojen kokoelma, jota yksi tai useampi tietojärjestelmä käyttää ja päivittää.

1 JOHDANTO

Tämän opinnäytetyön päätavoitteena on kehittää laitteisto, joka lukee ABB-robotin console- eli huoltosarjaportista saatavat tiedot muokaten ja tallentaen tiedot määriteltävään muotoon. Tallennettujen tietojen on oltava myös tallennettavissa tietokantaan.

1.1 Työn tausta

Tarve robottien tilatietojen (virheet, tilanmuutokset, varoitukset) tallentamiseksi ja seuraamiseksi etätiedonkeruuta käyttäen lähti robottien ylläpitäjän toiveesta saada tapahtuma- ja virheseuranta keskitetysti valvomolle ja sitä kautta helpommaksi.

Nykyisin robotin tietoja ei voi laajemmin tarkastella etäkäytöllä, vaan on aina mentävä kyseisen robotin ääreen ja tarkasteltava tietoja FlexPendant-käsiohjaimen avulla.

Toisena näkökohtana oli lähestyä tapahtumaseurantaa uudella tavalla, pyrkien kehittämään omanlainen edullinen, mukautuva ja lisenssitön tiedonkeruujärjestelmä.

1.2 Työn tavoitteet

Työn ensimmäisenä tavoitteena on selvittää robotin sarjaliikenteen toteutustapa sekä sieltä saatavilla oleva informaatio. Toisena tavoitteena on määritellä mitä tietoja ja minne ne tallennetaan sekä mitä muita mahdollisia toimintoja laitteella voisi olla. Lisätoimintoina voisi olla esimerkiksi kappale- ja käyntiaikalaskentaa. Lisäksi voidaan miettiä, miten tietojen siirto tietoa keräävän sekä etätietokoneen välillä voidaan toteuttaa pyrkien kuitenkin mahdollisuuksien mukaan yksinkertaiseen toteutukseen.

Lopuksi on vielä tarkoitus rakentaa toimiva protolaite ja testata laitteen toiminta lopullisessa toimintaympäristössä.

1.3 Työn rakenne

Opinnäytetyön ensimmäisessä luvussa kuvataan aiheen valintaa, työn tavoitteita, rakennetta sekä esitellään toimeksiantava yritys. Toisessa luvussa käsitellään opinnäytetyöhön kuuluvien ja siihen liittyvien laitteiden ja muiden asioiden yleistä teoriaa. Kolmannessa luvussa käsitellään suunnitteluprosessiin liittyviä asioita, esitellään järjestelmän kuvaus ja muita suunniteltuja asioita. Neljännessä luvussa käydään läpi käytännön toteutus. Viidennessä luvussa esitellään tulokset. Kuudennessa luvussa pohditaan projektin kulkua ja onnistumista sekä lopussa on lyhyt yhteenveto.

1.4 Toimeksiantajan esittely

Atria Oyj on kasvava ja kansainvälinen suomalainen elintarvikealan yritys. Atria toimii Pohjoismaissa, Venäjällä ja Baltian alueella (Atria [Viitattu 1.11.2014]).

Atrian liikevaihto oli vuonna 2013 1 411 miljoonaa euroa ja henkilökuntaa oli keskimäärin 4 668 henkilöä. Konserni on jakautunut neljään liiketoiminta-alueeseen. Liiketoiminta-alueet ovat Atria Suomi, Atria Skandinavia, Atria Venäjä ja Atria Baltia. (Atria [Viitattu 1.11.2014].)

Atrian asiakasryhmiä ovat elintarviketeollisuus, päivittäistavarakauppa ja Food Service -asiakkaat. Lisäksi Atrialla on omiin tuotemerkkeihin perustuvaa Fast Food -konseptiliiketoimintaa. (Atria [Viitattu 1.11.2014].)

Atrian juuret ulottuvat vuoteen 1903, jolloin perustettiin sen ensimmäinen omistajaosuuskunta. Atria Oyj:n osakkeet on listattu Nasdaq OMX Helsinki Oy:ssä. (Atria [Viitattu 1.11.2014].)

2 TEORIAA

Tässä luvussa esitellään opinnäytetyössä käytettävät laitteet, ohjelmat sekä järjestelmään liittyvät laitteet.

2.1 Teollisuusrobotti

Standardin (ISO 8373) mukaan teollisuusrobotti on uudelleen ohjelmoitavissa oleva monipuolinen, vähintään kolminivelinen mekaaninen laite, joka on suunniteltu liikuttamaan kappaleita, osia, työkaluja tai erikoislaitteita ohjelmoitavin liikkein monenlaisten tehtävien suorittamiseksi teollisuuden sovelluksissa. Uudelleen ohjelmoitavuus on siis olennainen piirre. (Lehtinen [Viitattu 8.11.2014, 2].)

IRB140 (kuvio 1) -tyypin 6-akselinen 6 kg:n hyötykuorman omaava teollisuusrobotti on suunniteltu erityisesti valmistusteollisuuteen. Robottia voidaan käyttää esimerkiksi maalaus-, hitsaus-, kokoonpano- ja pakkaussovelluksissa. (ABB [Viitattu 8.11.2014].)

Kuvio 1. ABB IRB 140 -robotti (ABB [Viitattu 8.11.2014]).

IRB 340 (kuvio 2) on 4-akselinen 2 kg:n hyötykuorman ja jopa 10 G:n kiihtyvyyden liikkeissään saavuttava teollisuusrobotti. IRB 340 -tyypin FlexPicker on omiaan esimerkiksi elintarviketeollisuuden nopeissa poiminta- ja pakkaussovelluksissa, kuten myös erilaisissa kokoonpanosovelluksissa. Tällä robotilla voidaan saavuttaa

jopa yli 150 poimintaliikettä minuutissa, riippuen liikematkasta ja kuormasta. (ABB 2012.)

Kuvio 2. ABB IRB 340 -robotti (ABB 2012).

Nykyään yhä yleisimmin robottien paikoitus- ja ohjaustoimintoja ohjataan erilaisten konenäkösovelluksien avulla (Kuivanen 1999, 56).

2.2 Ohjainyksikkö

Robotin ohjainyksikkö (kuvio 3), sisältää tarvittavan elektroniikan ohjaamaan robotin liikkeitä sekä oheislaitteiden toimintaa (ABB 2008a, 7).

Ohjainyksikkö sisältää eri moduuleita, kuten ohjausmoduulin, jolla ohjataan robotin akseleita eli liikkeitä. Lisäksi siinä on käyttömoduuli, joka sisältää tietokoneen, käyttäjäpaneelin, pääkytkimen, tiedonsiirtoliitännät, FlexPendant-liitännän, console -portin ja tilaa käyttäjän omille laitteille, kuten I/O-korteille. Ohjainyksikkö sisältää myös RobotWare-OS-käyttöjärjestelmän, jossa on kaikki perustoiminnot robotin toimintaan sekä ohjelmointiin. (ABB 2008a, 7.)

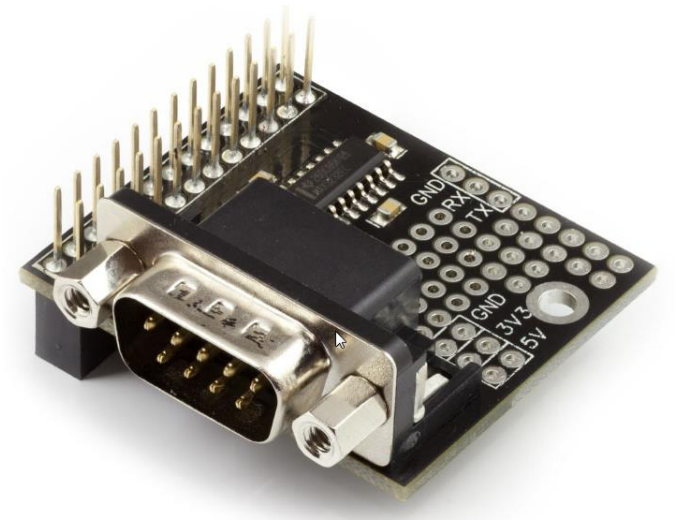
Kuvio 3. ABB IRC 5 -ohjainyksikkö (ABB [Viitattu 10.12.2014]).

2.3 Huoltosarjaportti

Console port eli huoltosarjaportti on robotin ohjainyksikössä sijaitseva sarjaportti, jonne ohjainyksikkö kirjoittaa eri tapahtumia. Tapahtumia ovat esimerkiksi käynnistyksen (boot) aikana ohjainyksikössä tapahtuvat asiat. Robotin virheet, varoitukset ja tilamuutokset tulostuvat myös huoltosarjaporttiin. Huoltosarjaporttiin kytketyllä tietokoneella ja terminaaliohjelmalla voi tarkastella robotin käynninaikaisia tapahtumia. (ABB 2008a, 68.)

2.4 RS232 TTL -sarjaliikenneadapteri

Rasberry Pi käyttää UART-portissa 3.3 V:n TTL-jännitetasoa. Raspberry Pin liittämiseksi sarjaporttiin tarvitaan sarjaliikenneadapteria (kuvio 4), joka voi perustua esimerkiksi Max3232-piiriin. Sarjaliikenneadapteri muuntaa TTL-jännitetason RS232-jännitetasoon sopivaksi. (AB Electronics UK [Viitattu 29.9.2014].)



Kuvio 4. Sarjaliikenneadapteri (AB Electronics UK [Viitattu 29.9.2014]).

2.5 Raspberry Pi

Raspberry Pi on Raspberry Pi -säätiön kehittämä edullinen noin luottokortin kokoinen tietokone (kuvio 5), jonka avulla säätiön tarkoitus on mahdollistaa kaikenikäisten ihmisten opiskelu ja tutustuminen tietotekniikkaan ja ohjelmointiin. Raspberry Pihin voidaan kytkeä televisio tai tietokoneen näyttö, näppäimistö, hiiri ja kaiuttimet. Raspberry Pihin voidaan kytkeytyä myös SSH-yhteyden avulla toisella tietokoneella, ilman edellä mainittuja ulkoisia laitteita. Raspberry Pitä voidaan käyttää useisiin samoihin tarkoituksiin kuin normaalia tietokonetta, esimerkiksi Internetin käyttöön, taulukkolaskentaan, tekstinkäsittelyyn ja peleihin. Raspberry Pi toistaa vaivattomasti myös Full HD -videota. (Raspberry Pi Foundation [Viitattu 10.11.2014].)

Raspberry Pin versio B+ julkaistiin heinäkuussa vuonna 2014. Raspberry Pi B+ on ohjelmallisesti alaspäin yhteensopiva aiempiin versioihin, vain laitteiston ominaisuuksia on lisätty ja paranneltu. Raspberry Pi on myös monipuolinen ja suuren suosion saavuttanut edullinen kehityskortti erilaisiin sulautettujen järjestelmien projekteihin sekä ohjelmoinnin opetteluun. Ohjelmointiin voi käyttää eri ohjelmointikieliä, kuten Scratch, Python, Java, C ja C++. (Raspberry Pi Foundation [Viitattu 11.11.2014].)

Raspberry Pi B+ sisältää useita erilaisia liityntöjä:

- microSD-muistikorttipaikka
- 4x USB 2.0 -liitin
- HDMI-liitin
- 4-nastainen 3,5 mm:n plugi-lähtö (audio- ja komposiittivideo)
- micro-usb-jänniteliitin
- RJ-45-Ethernet-liitin. (Raspberry Pi Foundation [Viitattu 11.11.2014].)

Laitteessa on myös 40-nastainen piikkirimaliitin, joka sisältää:

- GPIO-tuloja ja -lähtöjä
- UART-portin
- I^2C-väylän
- SPI-väylän
- 3.3 V:n ja 5 V:n jännitteet ja maapinnit. (Raspberry Pi Foundation [Viitattu 11.11.2014].)

Piirikortilta löytyvät liittimet myös Raspberry Pi:lle suunnitellulle kameramoduulille ja kosketusnäytölle (Raspberry Pi Foundation [Viitattu 11.11.2014]).

Kuvio 5. Raspberry Pi B+ (Raspberry Pi [Viitattu 15.10.2014]).

2.6 Käyttöjärjestelmä Raspbian

Raspbian on ilmainen Debian Wheezy Linuxiin pohjautuva, yksi monista Raspberry Pille optimoiduista käyttöjärjestelmistä. Käyttöjärjestelmään on sisällytetty monia hyödyllisiä perus- ja apuohjelmia. Lisäksi käyttöjärjestelmään on tehty yli 35 000 erilaista esikäännettyä ohjelmapakettia helposti asennettavaksi paketinhallinnan kautta. Raspbian-käyttöjärjestelmä on valmistunut vuonna 2012 ja on edelleen aktiivisen kehityksen alaisena. Raspbian ei ole mitenkään sidoksissa Raspberry Pi -säätiöön. Raspbian-käyttöjärjestelmää ovat kehittäneet Raspberry Pi -tietokoneeseen ja Raspberry Pi -säätiön opetuksellisiin tavoitteisiin mieltyneet kehittäjäryhmät. (Rasbian [Viitattu 29.9.2014].)

2.7 Geany-ohjelmaeditori

Geany on monipuolinen ohjelmaeditori useille käyttöjärjestelmille. Geany tukee monia tiedostotyyppejä, muun muassa C, C++, Java, PHP, HTML, Python, Perl ym. (Tröger ym. 2010.)

2.8 C++-ohjelmointikieli

C++ on yleiskäyttöinen järjestelmäohjelmointiin painottuva ohjelmointikieli, joka on kehitetty C-ohjelmointikielestä. C-ohjelmointikieli on siis C++-ohjelmointikielen kantakieli ja mahdollistaa näin myös C-kielen käyttämisen joitakin harvoja poikkeuksia lukuun ottamatta. (Stroustrub, 2000, 8, 13, 21, 847 - 858.)

C++ on monipuolinen C-ohjelmointikieli, joka tukee tiedon abstrahointia, olio-ohjelmointia ja geneeristä ohjelmointia (Stroustrub, 2000, 21).

2.9 Tietokanta

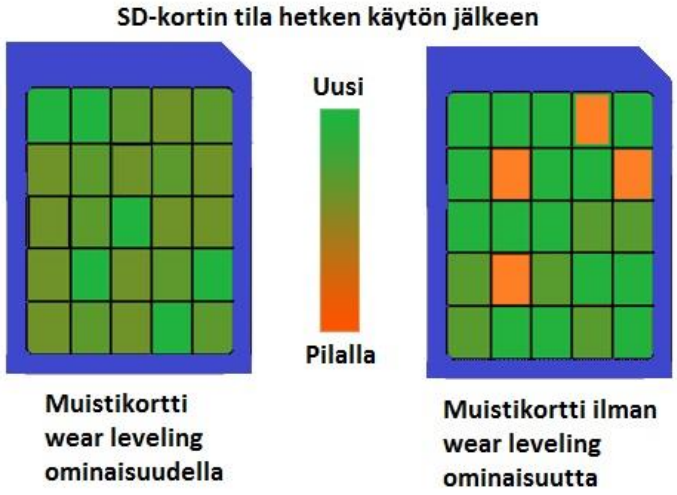
Yleisesti tietokannan voidaan sanoa olevan hallintajärjestelmä, jolla hallinnoidaan sinne tallennettuja loogisesti yhteenkuuluvia tietoja. Tietokannan tietoja voidaan hakea, muuttaa, päivittää ja lisätä esimerkiksi SQL-kyselykielen avulla. SQL-kyselykieli on saavuttanut lähes standardin aseman yleisimpänä tietokantakielenä. (Hovi, Huotari & Lahdenmäki 2003, 4, 5, 7, 8.)

Varhaisimmat tietokannan hallintajärjestelmät olivat hierarkkisia tai verkkomallisia. Nykyään lähes kaikki tietokantajärjestelmät perustuvat relaatiomalliin. Relaatiomalli on IBM:n tutkija E. F. Coddin v. 1970 kehittämä tietokantamalli, joka perustuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan. (Hovi, Huotari & Lahdenmäki 2003, 5, 7, 8.)

Tämän opinnäytetyön avulla kerättyjä tietoja on tulevaisuudessa tarkoitus tallentaa tietokantaan. Tietokantaohjelmistona on mahdollista käyttää esimerkiksi MySQL-tietokantaohjelmaa, joka on maailman yleisin Open Source -tietokantaohjelma. (Oracle [Viitattu 12.2.2015]). Tietokannasta haluttujen tietojen hakeminen olisi helppoa SQL-kyselykielen avulla. Esimerkiksi voitaisiin hakea vaikka määritellyt tapahtumat, tietyltä aikaväliltä laitekohtaisesti.

2.10 SD-muistikortti

Raspberry Pissä on SD-muistikortti käyttöjärjestelmän ja kaiken pysyvän tiedon tallentamiseen. Paljon muistikortille tallentavia sovelluksia ajettaessa on otettava huomioon myös muistikortin rajallinen kestoikä tallennuksien määrällä mitattuna. Tällaisia ovat esimerkiksi huomattavan paljon suuria tiedostoja käsittelevät sovellukset, sekä paljon tallentavat loggeri-sovellukset. Muistikortiksi on hyvä valita kortti, jossa on niin sanottu wear leveling -ominaisuus, joka ohjaa tallennustoimintoja tasaisesti muistikortin tallennuspinta-alalle (kuvio 6). (Freeman, 2014.)



Kuvio 6. SD-kortin täytön vertailu (Perustuu Freeman, 2014).

2.10.1 Tallennuksen teoriaa

Muistikortille tallennettaessa solu (cell) on ensin tyhjennettävä, ennen kuin siihen kirjoitetaan. Näin yhteen soluun tulee kaksi kirjoitusta jokaisella yhden bitin tallennuskerralla. Alkuperäinen flash-teknologia SLC (Single Level Cell) käyttää edellä mainittua tapaa. Muistin tallennuskapasiteetin kasvattamiseksi ja muistin hinnan pienentämiseksi on kehitetty MLC (Multi Level Cell) -tallennus, joka tallentaa kaksi bittiä yhteen soluun. Tällöin yhteen soluun tulee kolme kirjoitusta yhden bitin tallennuksen aikana (tyhjennys ja kaksi kirjoitusta). MLC-tallennustavalla on kuitenkin laskettu olevan noin 10-kertaa pienempi kestoikä, mitattuna P/E-sykleillä (kirjoitus / tyhjennys), verrattuna SLC-kortteihin. Kolmas tapa eli TLC (triple level cell) tallentaa kolme bittiä jokaiseen soluun. TLC-tallennuksen P/E-sykliluku on noin 1000,

joka tarkoittaa että muistikortti voidaan kirjoittaa vain noin 1000 kertaa täyteen ennen muistikortin vioittumista. SLC-korttien luku on noin 50 000 P/E-sykliä, MLC-korttien noin 5 000 P/E-sylkliä ja TLC-korttien noin 1000 P/E-sykliä. Tämä kertoo merkittävästä edusta SLC-muistikorttien hyväksi kirjoituskertojen määrän ja odotetun eliniän kannalta. Muistikortin tallennuksen hallinnan algoritmia ohjaa muistikortilla sisäänrakennettuna oleva mikrokontrolleri. (Kingston [Viitattu 4.2.2015, 3, 6].)

Mahdollisimman pitkän muistikortin kestoiän saavuttamiseksi on siis hyvä suosia SLC-muistikortteja, joissa on tallennuksen tasaus (wear leveling). Muistikortin kestoikää voi nostaa myös vähentämällä muut tallennukset minimiin. Raspberry Pin Linux-käyttöjärjestelmässä voi esimerkiksi poistaa käytöstä Swap-ominaisuuden (käyttömuistin laajennus muistikortille). Ensimmäinen keino tallennuksen tasauksen lisäksi on kuitenkin käyttää mahdollisimman suurta muistikorttia, jossa on paljon tyhjää tilaa. Suuri tyhjä tila mahdollistaa tallennuksien hajauttamisen suurelle pinta-alalle, kasvattaen näin tallennuskertojen määrää (kuvio 6). Joitakin muitakin hieman kyseenalaisia keinoja on vähentää käyttöjärjestelmän kirjoituksia muistikortille. (Stackexchange, 2012.)

Tallennuksen hallinta (wear leveling) on jaettu vielä kolmeen erilaiseen tapaan:

- Ei tallennuksen hallintaa
- Dynaaminen tallennuksen hallinta
- Staattinen tallennuksen hallinta. (SiliconSystems 2005.)

2.10.2 Ei tallennuksenhallintaa

Ilman tallennuksen hallinta-algoritmia oleva muistikortti kirjoittaa dataa aina samaan fyysiseen kohtaan muistikortilla. Esimerkkinä samaa 1 MB:n tiedostoa päivitettäessä poistetaan ja kirjoitetaan uudelleen 1800 kertaa tunnissa (= 30 kertaa minuutissa). Tämä antaa SLC muistikortin eliniän odotukseksi 2,3 päivää.

$$\frac{100\,000\ sykliä}{1800\ päivitystä\ tunnissa * 24\ tuntia} = 2{,}3\ \text{päivää}$$

(SiliconSystems 2005).

2.10.3 Dynaaminen tallennuksenhallinta

Dynaaminen tallennuksen hallinta-algoritmi on yleisin käytetty algoritmi. Tässä algoritmi hallitsee kortin vapaana olevaa tilaa. Otetaan esimerkiksi lasku, josta voi arvioida 4 GB:n SLC-muistikortin elinikää, kun siihen päivitetään edellisen esimerkin 1 MB:n tiedostoa 1800 kertaa tunnissa. Oletetaan, että lisäksi kortilla on tallennettuna käyttöjärjestelmä ja muuta staattista dataa 2 GB, joka ei muutu. Dynaaminen hallinta-algoritmi hallitsee siis tässä tapauksessa 50 % muistista. Muistikortin eliniän laskenta P/E-syklien mukaan laskettuna voidaan suorittaa seuraavalla kaavalla:

$$\text{Kestävyys (vuotta)} = \frac{(C-D)*E*(1-M)}{(S*f)*525600\frac{min}{vuosi}} \qquad (1)$$

Jossa:

C = muistikortin koko MB (MB = GB*1024)
D = staattisen datan määrä kortilla (MB), esim. käyttöjärjestelmä
E = muistikortin P/E-luku (SLC-kortilla 100 000)
M = turvamarginaali
S = kirjoitettavan tiedoston koko MB (KB = MB*1024)
f = kirjoitustaajuus minuutissa
eli

$$\frac{(4096MB - 2048MB) * 100\,000\ sykliä * (1 - 0.05)}{\left(1MB * 30\frac{kirjoitusta}{min}\right) * 525600\ \frac{min}{vuodessa}} = 12{,}3\ vuotta$$

(SiliconSystems 2005.)

2.10.4 Staattinen tallennuksenhallinta

Staattinen tallennuksen hallinta-algoritmi taas hallitsee koko muistikortin tallennustilaa. Algoritmi etsii vähiten tallennettuja fyysisiä alueita, havaitessaan niissä staattista dataa se siirtää sitä alueelle, jolla on suurin määrä tallennuskertoja. Näin dynaaminen data saa koko ajan uutta tilaa ja muisti kiertää, kunnes koko muistikortin

solut on kirjoitettu maksimiarvoihin ja muisti muuttuu lopulta käyttökelvottomaksi. Staattinen hallinta-algoritmi hidastaa hiukan muistiin kirjoittamista, mutta antaa vastineeksi maksimoidun käyttöiän. Edellisen esimerkkilaskun arvoilla 4 GB:n SLC-muistin ikä staattisella muistinhallinnalla on noin 24,7 vuotta.

$$\frac{(4096MB - 0MB) * 100\,000\ sykliä * (1 - 0.05)}{\left(1MB * 30\frac{kirjoitusta}{min}\right) * 525600\ \frac{min}{vuodessa}} = 24{,}7\ vuotta$$

(SiliconSystems 2005.)

2.10.5 Yhteenveto

Laskukaavoista näkee, että jakoviivan alla olevien tallennettavan tiedoston koon ja tallennuskertojen määrän kasvaessa oletettu kestoikä kirjoitus / lukukertojen (P/E) määrällä mitattuna putoaa nopeasti. Esimerkin SLC-muistikortin elinikä dynaamisella tallennuksenhallinnalla huomattavan paljon tallentavissa Raspberry Pi -sovelluksissa voidaan laskea muutamissa vuosissa (<10 vuotta). Halvemmilla ja yleisemmillä MLC-tyypin korteilla P/E-luku on vielä noin 10 kertaa pienempi, verrattuna esimerkeissä käytettyyn SLC-tyypin muistikorttiin. Tällöin myös eliniän odotus on vastaavasti 10 kertaa pienempi. Ilman tallennuksen hallintaa olevat kortit eivät sovellu teollisiin dataa tallentaviin sovelluksiin ollenkaan huomattavan lyhyen elinikäennusteen vuoksi.

Raspberry Pi -sovelluksissa muistikortin elinikää laskee vielä lisäksi Linux-käyttöjärjestelmän journaloiva eli kirjaa pitävä tiedostojärjestelmä ja monet loki-, swap- ym. tallennukset. Yleisimmin käytössä olevilla MLC-tyypin SD-muistikorteilla elinikäarvio olisi vain noin yksi vuosi, edellisten esimerkkilaskujen perusteella.

Taulukossa 1 on havainnollistettu eri Flash-tekniikoiden elinikäennuste, perustuen edellä oleviin esimerkkilaskuihin 4 GB:n muistikortilla, sekä esitettyjen P/E-syklien suhteisiin eri tallennuksenhallintatavoilla. Muistikortin koon kaksinkertaistaminen vähintäänkin kaksinkertaistaa elinikäennusteen muistinhallintaa käyttävillä muistikorteilla.

Taulukko 1. Tallennuksen hallinnan vaikutus muistin elinikään

Flash-teknologia	Ei tallennuksen hallintaa	Dynaaminen tallennuksenhallinta	Staattinen tallennuksenhallinta
SLC	2,3 päivää	12,3 vuotta	24,7 vuotta
MLC	0,23 päivää	1,23 vuotta	2,47 vuotta
TLC	0,023 päivää	0,25 vuotta	0,49 vuotta

2.11 UPS-varajännitesyöttö

Raspberry Pi -tietokoneen katkeamaton käyttöjännitesyöttö (UPS) on tärkeää varmistaa, koska laitteen tiedostojärjestelmä voi vioittua äkillisissä jännitekatkoksissa. Etenkin teollisissa käyttökohteissa on erittäin suotavaa varmistaa katkeamaton jännitesyöttö käyttövarmuuden maksimoimiseksi sekä huoltotoimenpiteiden minimoimiseksi.

2.12 Robotin tapahtumat

Robotin ohjainyksikkö pitää yllä tapahtumalokia, joka sisältää robotissa tapahtuneita yksilöityjä virhe-, varoitus- ja tilanmuutosilmoituksia (ABB 2008b, 83). Tapahtumalokia voi tarkastella paikallisesti FlexPendant-käsiohjaimella. Ohjainyksikkö kirjoittaa myös samat tapahtumat ohjainyksikön huoltosarjaporttiin. Huoltosarjaportista virheilmoituksia voi seurata esimerkiksi Putty-terminaaliohjelman avulla määrittäen sarjaportin asetuksiksi 9600,8,N.

Robotin tapahtumailmoituksen muoto voi olla esimerkiksi,

14-09-14 11:46:27 MC0: type = ERROR id = System code = 205.

Virheilmoituksesta selviää virheen päiväys ja kellonaika, milloin virhe on tapahtunut. MC0: type kertoo tapahtuman tyypin: tilanmuutos, varoitus tai virhe. Id kertoo

mihin robottijärjestelmän osaan tai piirteeseen tapahtuma kuuluu ja code kertoo tapahtuman koodinumeron. (ABB 2008b, 85 - 91.)

Robotin tapahtumaviestistä voidaan muodostaa viisinumeroinen koodi. Edellisen järjestelmätapahtumiin kuuluvan virheilmoituksen koodi on 20205. Koodi muodostuu summaamalla id-kategorian tapahtumasarjalukuun code-numero, eli 20000 + 205 = 20205. Tämän koodin perusteella ohjainyksikön käyttäjän oppaan vianmääritysosiosta löytyy kattavampi selite tapahtumalle. Käyttäjän oppaan vianmääritysosiossa on esiteltynä karkeasti laskien noin 3000 kappaletta erilaista tapahtumaa. Taulukossa 2 on taulukko tapahtumanumerosarjoista. (ABB 2008b, 83 - 91, 161 - 513.)

Taulukko 2. Ohjainyksikön tapahtumanumerosarjat (Perustuu ABB 2008b, 85).

Numerosarja	Tapahtuman tyyppi
1x xxx	Käyttötapahtumat, järjestelmän käsittelyyn liittyvät tapahtumat
2x xxx	Järjestelmätapahtumat, jotka liittyvät järjestelmän toimintoihin ja tiloihin
3x xxx	Laitteistotapahtumat, jotka liittyvät järjestelmän laitteistoon
4x xxx	Ohjelmatapahtumat, jotka liittyvät RAPID-käskyihin, tietoihin jne.
5x xxx	Liiketapahtumat, jotka liittyvät käsittelijöiden liikkeiden ja asemien hallintaan
7x xxx	I/O-tapahtumat, jotka liittyvät tuloihin ja lähtöihin, tiedonsiirtoväyliin jne.
8x xxx	Käyttäjätapahtumat, käyttäjän määrittämät tapahtumat.
11 xxxx	Prosessitapahtumat; sovelluskohtaiset tapahtumat
12 xxxx	Kokoonpanotapahtumat, järjestelmän kokoonpanoon liittyvät tapahtumat
13 xxxx	Maalaus

Tässä opinnäytetyössä on pääosassa juuri näiden tapahtumailmoitusten lukeminen, muokkaaminen ja tallentaminen tietokannan saataville.

3 SUUNNITTELUPROSESSI

Tässä luvussa kuvaillaan erilaisia tässä opinnäytetyössä tarvittavia määritelmiä, suunnitelmia ja kuvauksia, miten järjestelmä on rakennettu.

3.1 Yleistä

Opinnäytetyön aiheeksi on valittu ABB:n robotin huoltosarjaportista luettavissa olevien tapahtumien tallentaminen. Tallennettujen tietojen olisi oltava luettavissa etäkäytöllä ja siirrettävissä tietokantaan. Tietokantaosuus ei kuulu tämän työn aihepiiriin, muuten kuin luomalla tietokannalle lähde, mistä tietokanta saa tiedot noudettua.

Aiheen selvittyä syksyn 2014 aikana, suunniteltiin alustavasti miten työ toteutetaan ja minkälaisin laittein. Syyskuussa 2014 koulun hyväksyttyä tutkimussuunnitelman, tarvittavat laitteet ja osat tilattiin ja kerättiin kokoon opinnäytetyötä varten.

Laitteiden kokoamisen jälkeen aloitettiin ohjelman suunnittelu ja ohjelmointi. Ohjelmaa testattiin aina kunkin ohjelman osan yhteydessä pieninä kokonaisuuksina.

Ensimmäinen ja tärkein välitavoite oli selvittää, minkälaista dataa robotin huoltosarjaportista saadaan, sekä lukea ja tallentaa data Raspberry Pin sarjaportin kautta tiedostoon. Välitavoitteen selvittämiseksi luettiin huoltosarjaporttia ensin kannetavan tietokoneen ja terminaaliohjelman avulla jonkin aikaa. Tällä tavoin saatiin huoltosarjaportista saatavasta datasta näytetiedosto tallennetuksi. Tämän jälkeen ohjelmoitiin yksinkertainen sarjaportin lukuohjelma, joka tallensi kaiken tiedon sellaisenaan Raspberry Pi -tietokoneen tiedostoon. Lukuohjelmaa testattiin molemmissa ohjainyksikkötyypeissä ja huomattiin, että vanhemmassa ohjainyksikössä jokaisen rivin alkuun tulostui kaksi ylimääräistä kontrollimerkkiä, mitkä olisi poistettava jatkossa. Muuten todettiin, että sarjaportin lukeminen ja tiedostoon tallennus onnistuu, tästä on hyvä jatkaa suunnittelua.

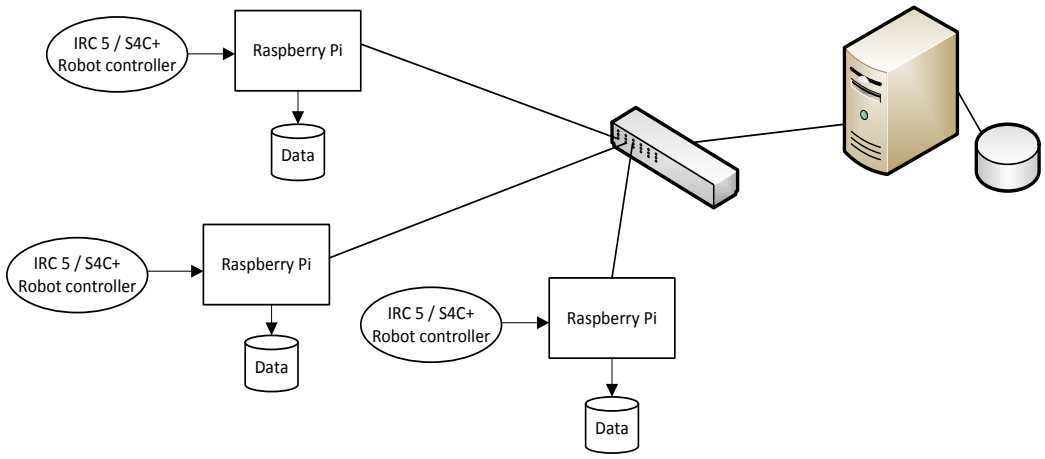
Ensimmäisen välitavoitteen selvittyä jatkotestausta ohjelman eri vaiheista voidaan suorittaa kytkemällä Raspberry Pi -tietokone toiseen tietokoneeseen sarjakaapelilla ja emuloimalla terminaaliohjelmalla tiedon syöttöä Raspberry Pille. Aikataulullisesti opinnäytetyö oli tarkoitus saada valmiiksi huhtikuun (2015) aikana.

3.2 Järjestelmän kuvaus

Järjestelmään kuluu robottikohtaisesti seuraavat laitteet:

- Raspberry Pi B+ -tietokone + microSD-kortti (8 GB)
- sarjaportin adapteri TTL-tasojen sovittamiseksi
- nollamodeemisarjaliikennekaapeli
- Raspberry Pi -virtalähde
- pieni UPS-varajännitelähde
- yhteinen tietokantapalvelin (lähitulevaisuudessa).

Robotteja on yrityksessä useita ja ne yhdistetään omassa Ethernet-verkossa, erillään yrityksen muusta verkosta, kytkinten välityksellä tietokannan sisältävälle valvomo PC:lle (kuvio 7).



Kuvio 7. Järjestelmän lohkokaavio.

3.3 Robotin tapahtumien tallennus

Järjestelmä tallentaa robotin tapahtumia itsenäisesti. Järjestelmä lukee, muokkaa ja tallentaa tapahtumat tiedostoon, tietokannan saataville. Tiedostomuotona käytetään CSV-tiedostomuotoa. CSV-tiedostomuodolla voidaan tallentaa yksinkertaista taulukkomuotoista tietoa tekstitiedostoon. Järjestelmä muodostaa luettavista tapahtumista yksilöllisen viisinumeroisen koodin. Koodi on robotin ohjainyksikön käyttäjän oppaan vianmääritysosion virheluettelon mukainen.

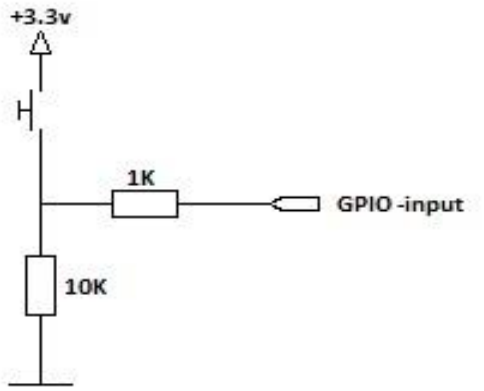
Tapahtumien muokkaamisella tarkoitetaan seuraavia asioita:

- tyhjien rivien tunnistus ja poisto
- viestin alussa olevien ylimääräisten merkkien tunnistus ja poisto (S4C+ -ohjainyksikkö)
- alkuperäisen aikaleiman poisto (robotin kello ja päiväys voi olla asettamatta oikeaan)
- uuden aikaleiman talletus Raspberry Pin ajasta (kaikissa sama, aikapalvelimelta noudettu aika)
- virhekoodin muodostus tapahtuman tiedoista
- järjestysnumeron muodostus
- laitteen yksilöivän ID-numeron haku asetustiedostosta
- tapahtumaviestin tietojen eriytys

- uuden tallennettavan tapahtumarivin luonti em. tiedoista ja erotusmerkkien lisäys (tietokanta varten)
- kaikkien tietojen tallennus tiedostoon CSV-tiedostomuotoon

3.4 Digitaalitulot

Järjestelmässä käytössä olevat tulot kytketään käyttöön alasvetovastusten avulla. Ilman ylös- tai alasvetovastuksien käyttöä tulot kelluvat määräämättömästi ja voivat räpsytellä tuloa päälle tai pois hallitsemattomasti. Alasvetovastus vetää tulon nollapotentiaaliin, jolloin tulon tila on laitteessa LOW, kun kytkintä ei ole painettu (kuvio 8). Tuloon kytketyn kytkimen painaminen kytkee 3.3 voltin käyttöjännitteen tuloon, jolloin tulon tila laitteessa on HIGH. Kuviossa 8 on kuvattu käytetty alasvetokytkentä. Kytkennässä (kuvio 8) 10 KΩ:n vastus vetää tulon nollapotentiaaliin. Toinen 1 KΩ:n vastus suojaa tuloa kytkintä painettaessa suoralta oikosululta, jos ohjelmassa on tulo virheellisesti määritelty lähdöksi.



Kuvio 8. Tulon alasvetokytkentä

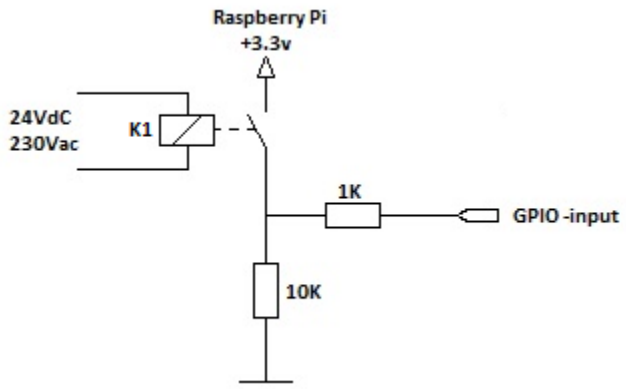
Taulukossa 4 on esitetty Raspberry Pin I/O-kytkentäliittimet. Taulukosta käy ilmi fyysisen kytkentäliittimen numerointi, liittimen toimintatapa, sekä WiringPi-kirjaston käyttämä numerointitapa liittimille. Samaan taulukkoon on liitetty myös tämän opinnäytetyön I/O-pisteluettelo. Liittimiin on mahdollista kytkeä myös erilaisia väyliä, esimerkiksi I²C ja SPI, joita voi tarvittaessa käyttää myös normaaleina I/O-

pisteinä. Nämä erikoisliittimet on hyvä jättää käyttämättä, koska väyliä voidaan myöhemmin tulla tarvitsemaan.

Taulukko 3. I/O-pisteluettelo.

IO	WiringPi	Mode	Physical Header Pin		Mode	WiringPi	IO
		3.3 V	1	2	5 V		Ei saa kytkeä tuloihin
	8	SDA/I2C/GPIO2	3	4	5 V		Ei saa kytkeä tuloihin
	9	SCL/I2C/GPIO3	5	6	GND		Sarjaliikenne adapteri
	7	GPIO4	7	8	TxD/GPIO14	15	Sarjaliikenne adapteri
		GND	9	10	RxD/GPIO15	16	Sarjaliikenne adapteri
	0	GPIO17	11	12	GPIO18	1	
	2	GPIO27	13	14	GND		
	3	GPIO22	15	16	GPIO23	4	
		3.3 V	17	18	GPIO24	5	
	12	MOSI/GPIO10	19	20	GND		
	13	MISO/GPIO9	21	22	GPIO25	6	
	14	SCLK/GPIO11	23	24	CE0/GPIO8	10	
		GND	25	26	CE1/GPIO7	11	
	varattu	ID_SD	27	28	ID_SC	varattu	
	21	GPIO5	29	30	GND		
IN_Käyntiaikalaskuri_1	22	GPIO6	31	32	GPIO12	26	IN_Kappalelaskuri_1
IN_Käyntiaikalaskuri_2	23	GPIO13	33	34	GND		
OUT_RPi_käy	24	GPIO19	35	36	GPIO16	27	IN_Kappalelaskuri_2
OUT_varalla	25	GPIO26	37	38	GPIO20	28	IN_Verkkojänniteen tila
		GND	39	40	GPIO21	29	IN_varalla

3.4.1 Ulkoisen tiedon kytkeminen tuloon



Kuvio 9. Ulkoisen tiedon kytkentä digitaalituloon.

Ulkoisen tiedon kytkeminen laitteen digitaalituloon on esitetty kuviossa 9. Ulkoiset tulot on syytä kytkeä releen (K1) välityksellä. Tällä tavoin Raspberry Pi -laitteen tulo on paremmin suojattuna ulkoisilta yli- ja häiriöjännitteiltä. Releen kelajännit-

teen valinnalla voidaan määritellä useita erilaisia jännitteitä käyttävien laitteiden tilatietoja, välitettäväksi Raspberry Pi -tietokoneen tuloon.

3.5 Ohjelmointikielen valinta

Ohjelmointikieleksi valittiin C++-ohjelmointikieli toimeksiantajan edustajan ja ohjaavan opettajan ehdotuksesta. C- ja C++-ohjelmointikielet ovat yleisiä ohjelmointikieliä erilaisissa sulautettujen järjestelmien laitteissa. C++-ohjelmointikielestä on myös saatavilla runsaasti kirjallisuutta ja muita lähteitä ohjelmointikielen opiskeluun.

3.6 Talletettavien tietojen määritys

Tiedot tallennetaan kahteen tiedostoon, huoltosarjaportin tapahtumatiedostoon ja laskurien tiedostoon. Tallennettavien tietojen erotusmerkkinä käytetään puolipistettä. Puolipisteen käytön avulla saadaan tiedosto tallennettua CSV-tallenusmuotoon. Tallennettavan tapahtumarivin alkuun lisätään aina kaikissa tapauksissa seuraavat kentät.

- järjestysnumero
- päivänmäärä
- laitteen yksilöivä ID-numero
- päiväys ja kellonaika (vain huoltosarjaportin lukutiedosto)
- päiväys ja tallennustunti (vain laskurien tiedosto).

Huoltosarjaportista luettujen ja muiden järjestelmän tapahtumien tietojen tallennus ja muokkaus tapahtuu aina, kun tapahtuma ilmaantuu. Robotin muokattavat tapahtumailmoitukset ovat muodoltaan seuraavanlaisia.

14-09-01 11:46:25 MC0: type = STATE_CHANGE id = Operational code = 12

tai esimerkiksi seuraavanlaisia erilaisia tekstirivejä ilman päiväystä ja aikaleimaa.

arg 1: /MainModule/VieTyhjaVasen/MoveJ/315

Eriytettynä tietokenttinä tapahtumailmoituksista tallennetaan:

- muodostettava virhenumero
- MC
- TYPE
- ID (tapahtumaviestin ID-kenttä)
- CODE
- MESSAGE.

Lisäksi tallennetaan ilman aikaleimaa olevat robotin tapahtumailmoitukset MESSAGE-kenttään. Viimeisenä kenttänä tallennetaan Linux-järjestelmän käynnissäoloaika tunteina ja minuutteina (katso liitteet 3 - 6).

Ohjelmoitujen laskureiden osalta tallennetaan tiedostoon laskureiden arvot tunneittain.

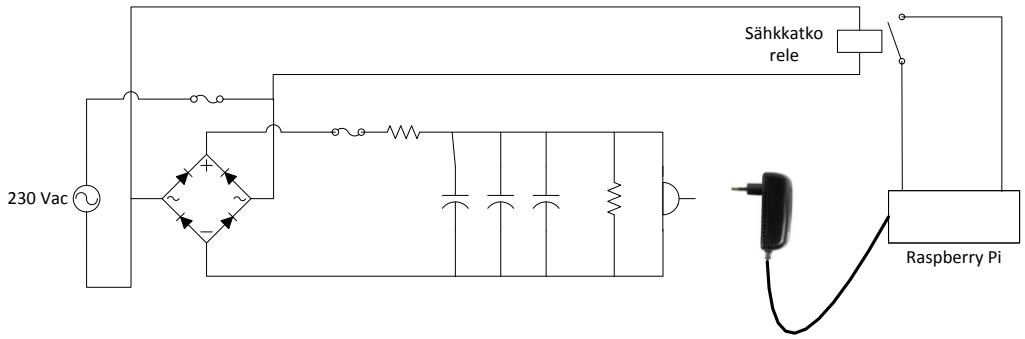
- kappalelaskenta 1
- kappalelaskenta 2
- käyntiaika 1 (sekuntia)
- käyntiaika 2 (sekuntia).

3.7 UPS-varajännitelähde

UPS-jännitteen syötölle on olemassa kaksi perusperiaatteeltaan erilaista mahdollisuutta, akkuihin tai kondensaattoreihin perustuvat varajännitelähteet. Akkujen etuna on pitempi jännitteensyöttöaika (tunteja), riippuen akun varauskyvystä. Haittana on akkujen kunnon seuranta ja ajoittainen vaihtotarve. Kondensaattori toteutuksessa etu on huoltovapaus ja haittana pienempi jännitteensyöttöaika (minuutteja).

Raspberry Pille on saatavissa räätälöityjä molempiin periaatteisiin perustuvia varajänniteratkaisuja. Yksi mahdollisuus on myös perinteinen pienikokoinen verkkovirta-UPS-laite, joita on saatavilla edullisesti (<100 €), mikä on ehkä edullisin keino jos lähekkäin on useampi Raspberry Pi -laite. Usein yrityksissä on myös isompi yleinen UPS-varajänniteverkko erilaisille laitteille ja tietokoneille, minkä hyödyntämistä voisi myös harkita, mikäli sellainen on lähettyvillä helposti saatavilla.

Tässä opinnäytetyössä haluttiin kuitenkin etsiä mahdollisimman huoltovapaa ratkaisu varajännitesyötölle. Eri vaihtoehtoja vertailtaessa kondensaattoreilla toteutettu ratkaisu osoittautui toimivaksi ratkaisuksi. Reunaehtojen selvittyä päädyttiin rakentamaan koekäyttöön ratkaisu, joka on käyttövarma ja täysin huoltovapaa.



Kuvio 10. Rakennetun UPS-jännitesyötön periaatekuva.

Ratkaisussa verkkojännite ensin tasasuunnataan ja tähän tasajännitteeseen lisätään tarvittava määrä kondensaattoreita halutun varajännitteensyöttöajan saavuttamiseksi. Muodostetulla tasajännitteellä syötetään normaalia hakkurivirtalähdettä. Piirissä suojalaitteina ovat sulakkeet, syöksyvirtaa rajoittava vastus sekä purkuvastus kondensaattoreille (kuvio 10).

Hakkurivirtalähde toimii myös tasajännitesyötöllä, vaikka siihen on leimattu tulojännitealue 100 Vac - 240 Vac. Hakkurissa syöttävä vaihtojännite tasasuunnataan myös itsessään, joten käytännössä siihen voi syöttää myös vastaavasti muodostettua tasajännitettä. Lisäksi laaja syöttöjännitealue edesauttaa tuottamaan tarvittavaa varajännitettä kondensaattoreiden avulla.

Raspberry Pi B+:n mitattu virrankulutus ilman lisälaitteita oli noin 300 mA. Tarvittavien kondensaattorien määrä on laskettu 1 ampeerin tehonkulutusarviolla noin 1 minuutin varajännitteen syöttöajalle. Lisäksi Raspberry Pi GPIO -tuloliittimeen kytketään erillisen sähkökatkoreleen kärkien kautta tuleva tieto, jota käytetään tulkitsemaan verkkojännitteen tila.

Toiminta sähkökatkossa on seuraava: sähkön ollessa yli 30 sekuntia yhtäjaksoisesti pois päältä Raspberry Pissä oleva C++-kielinen ohjelma antaa Linux-käyttöjärjestelmälle käskyn

shutdown -h now

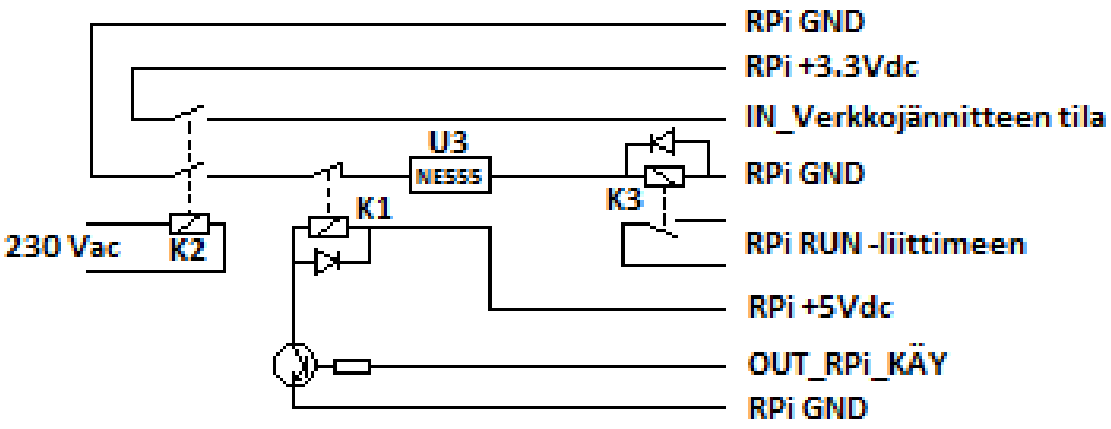
ja sammuttaa laitteen hallitusti. Raspberry Pin sammuminen kestää noin kymmenen sekuntia. Kondensaattoreiden varajännitekapasiteettia ei ole tarkoitus käyttää täysin loppuun. On otettava huomioon mahdolliset tehonkulutuspoikkeamat. Laite sammuu siis laskennallisesti 20 sekuntia ennen kondensaattoreiden tyhjentymistä.

Haittapuolena tämän UPS-laitteen toiminnassa on, että Raspberry Pi ei välttämättä käynnisty uudelleen automaattisesti. Raspberry Pi ei käynnisty automaattisesti uudelleen, mikäli verkkojännite ei ole poikki niin kauan että kondensaattorit tyhjentyvät riittävästi ja katkaisevat käyttöjännitteen Raspberry Pi -laitteelta.

Ensiratkaisuna tämän voi kiertää kytkemällä verkkojännitepuolelle vetohidastetun aikareleen, joka katkaisee verkkojännitesyötön aina sähkökatkossa. Aikareleen vetohidastusajan on oltava pitempi kuin kondensaattorien purkautumiseen kuluva aika. Tässä ratkaisussa Raspberry Pi -laite joudutaan sammuttamaan aina sähkökatkon aikana. Sammutus tapahtuu kuitenkin hallitusti. Kondensaattoreiden varajännitteen ylläpitoaikaa voi lyhentää kattamaan vain pelkkään sammutukseen kuluvan ajan. Näin Raspberry Pi -laite lähtee aina käyntiin sähkökatkon jälkeen, koska kondensaattorit ehtivät tyhjentyä ennen kuin vetohidastettu aikarele kytkee verkkojännitteen jälleen UPS-piiriin.

Toinen, parempi tapa ratkaista automaattinen käynnistyminen on tarkkailla verkkojännitteen ja Raspberry Pi -tietokoneen tilaa. Kuviossa 11 on esitetty periaatekuva ratkaisusta ja liitteessä 1 on tarkempi piirikuva ajastimen kytkennästä. Opinnäytetyössä suunniteltiin ja rakennettiin tähän ratkaisuun perustuva testikytkentä ja testattiin toiminta onnistuneesti. Kytkentä tutkii Raspberry Pin ja verkkojännitteen tilaa. Kytkentä käynnistää Raspberry Pi -tietokoneen antamalla RUN-liittimeen lyhyen käynnistyspulssin, jos sähkökatkon jälkeen verkkojännite on palautunut ja Raspberry Pi on sammuneena. Edellä mainittujen käynnistysehtojen ollessa voimassa voidaan lyhyt käynnistyspulssi suorittaa NE555-piirillä toteutetun ajastinkytkennän ohjaaman releen potentiaalivapaan koskettimen avulla. Tällä tavoin Rasp-

berry Pi -laitteen uudelleen käynnistyminen varmistetaan tilanteessa, jossa verkkojännite palautuu ennen kondensaattorien tyhjentymistä. Näin laitteen ei tarvitse sammuttaa itseään sähkökatkoissa, jotka kestävät ohjelmassa säädettyä 30 sekuntia lyhyemmän ajan.



Kuvio 11. Raspberry Pin käynnistinlogiikan periaatekuva

Tämän UPS-laitteen käyttöönotossa on huomioitava voimassa olevat sähköturvallisuusmääräykset asennuksen, koteloinnin ja käytön osalta, koska laite sisältää hengenvaarallisen jännitteen.

3.8 Piirilevy

Laitteelle suunniteltiin ja rakennettiin myös protopiirilevy laitteen ulkopuolisia liityntöjä ja toimintoja varten. Piirilevyssä on liitettäville tuloille

- riviliittimet
- alasveto- ja suojavastukset (katso kappale 3.4)
- käytettävälle lähdölle transistori ja rele.

Piirilevy liitetään Raspberry Pi -tietokoneeseen lattakaapelin avulla. Piirilevyssä on myös kappaleessa 3.7 kuvattu verkkojännitteen ja Raspberry Pi -tietokoneen tilojen tarkkailu- ja käynnistyslogiikka. Logiikan avulla voidaan Raspberry Pi -laite käynnistää automaattisesti sähkökatkosta johtuneen sammutuksen jälkeen.

Laitteet oli suunniteltu sijoitettavan kompaktiin koteloon, jossa UPS-kondensaattorit ja verkkojännitteet on vielä erikseen eristettynä muista laitteista.

4 KÄYTÄNNÖN TOTEUTUS

Tässä luvussa kuvaillaan opinnäytetyön käytännön toteutusta käyttöjärjestelmän asennuksesta, sekä muista alkumäärittelyistä ja ohjelman toiminnankuvauksesta aina uuden laitteen perustamiseen.

4.1 Alkumäärittelyt

Uutta Raspberry Pi -moduulia perustettaessa on ensin tehtävä tiettyjä tässä kappaleessa kuvailtuja asennus- ja määrittelytoimenpiteitä. Nämä toimenpiteet tarvitsee kuitenkin tehdä vain kerran. Kaikkien asennusten ja määritysten ollessa oikein Raspberry Piin muistikortista otetaan levykuva (image). Levykuva voidaan ottaa esimerkiksi Win32DiskImager-ohjelmalla. Tämä levykuva toimii varmuuskopiona ja mahdollistaa seuraavien moduulien perustamisen helposti ja nopeasti.

4.1.1 Käyttöjärjestelmän asennus

Tässä opinnäytetyössä käyttöjärjestelmänä käytetään Rasbian Wheezy Linux -käyttöjärjestelmää, joka asennetaan Raspberry Pin muistikortille. Laitteita tilattaessa tilattiin myös 8 GB:n muistikortti, joka sisältää esiasennetun NOOBS (New Out Of the Box Software) -käyttöjärjestelmän asennushallinnan. Asennus käynnistyy ohjatusti asettamalla muistikortti Raspberry Pin muistikorttipaikkaan ja kytkemällä näppäimistö, hiiri, sekä jännite USB-virtalähteellä laitteeseen. NOOBS sisältää useamman eri käyttöjärjestelmän asennusvaihtoehdon erilaisiin käyttötarkoituksiin, tässä valittiin Rasbian. Asennuksen valmistuttua on hyvä päivittää pakettienhallinan lista ja asennetut paketit ajan tasalle komennoilla:

sudo apt-get update

sudo apt-get upgrade

Käyttöjärjestelmiä voi ladata myös suoraan internetistä ilmaiseksi. (Raspberry Pi Foundation [Viitattu 10.3.2015].)

4.1.2 Laitteiston kellonaika ja päiväys

Raspberry Pi ei sisällä reaaliaikakelloa ja tästä syystä kellonaika ja päiväys ovat väärässä laitteen käynnistymisen jälkeen. Normaalisti Raspberry Pi hakee päiväyksen internetissä olevilta yleisiltä aikapalvelimilta. Tässä tapauksessa Raspberry Pi ei ole kytkettynä yleiseen internetverkkoon, joten on oltava keino miten laite saadaan oikeaan aikaan käynnistymisen yhteydessä. Ratkaisuna Raspberry Pi määritellään hakemaan kellonaika ja päiväys valvomotietokoneella olevalta aikapalvelimelta käynnistyessään. Määritys tehdään */etc/ntp.conf*-tiedostossa lisäämällä aikapalvelimen IP-osoite.

Oikea päiväys ja kellonaika on tärkeää saada, koska sitä käytetään ohjelmassa tapahtumaviestien aikaleiman lähteenä.

4.1.3 Sarjaportin konsoliominaisuuden poisto

Raspberry Pi käyttää oletuksena laitteen sarjaporttia konsolina sarjaportin GPIO-liittimissä 8 ja 10. Oletusmäärityksellä Raspberry Pi -laitteeseen voi kytkeytyä ja käyttää laitetta terminaaliyhteyden avulla. Konsolimääritys on konfiguroitava pois käytöstä, jos aikomuksena on käyttää sarjaporttia ohjelmissa, kuten tässä työssä. Muutos suoritetaan muokkaamalla /etc/inittab- ja /boot/cmdline.txt-tiedostoista konsoliominaisuus pois käytöstä. (AB Electronics UK [Viitattu 10.3.2015].)

Cmdline.txt-tiedostosta poistetaan kaikki *ttyAMA0*-viittaukset ja inittab-tiedostossa kommentoidaan seuraava rivi #-merkillä:

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

(AB Electronics UK [Viitattu 10.3.2015]).

4.1.4 Samba

Samban asennus tarvitaan tiedostojaon sallimiseksi Windows-käyttöjärjestelmän kanssa. Tiedostojaon kautta valvomotietokoneelta voidaan lukea jaetun hakemiston sisältöä ja kopioida esimerkiksi mahdollinen C++-päivitystiedosto.

Samban asennus on helpointa suorittaa päätteessä komennolla:

sudo apt-get install samba samba-common-bin

Asennuksen jälkeen on luotava jaettava hakemisto komennolla:

sudo mkdir -m 777 /home/pi/robot

Lisäksi määritellään asiaan kuuluvia asetuksia */etc/samba/smb.conf*-asetustiedostossa. Ennen asetustiedoston muokkaamista siitä on hyvä tehdä varmuuskopio komennolla:

cp /etc/samba/smb.conf /etc/samba/smb.conf.bak

Asetustiedostoon muutetaan seuraavat kohdat:

workgroup = workgroup (työryhmän nimi)
wins support = yes

Seuraavat rivit lisätään tiedoston loppuun ”share definitions” -alueelle:

[robot]
comment = Data share
path = /home/pi/robot (jaettava hakemisto)
browseable = yes
read only = no
create mask=0777
directory mask=0777

Asetustiedoston muokkaamisen jälkeen on vielä hyvä testata asetustiedoston asetusten oikeellisuus komennolla:

testparm

Mikäli virheitä ei asetustiedostosta löydy, voidaan Samba käynnistää uudelleen komennolla:

sudo service samba restart

Uudelleenkäynnistyksen jälkeen tiedostonjako on käytössä.

4.1.5 WiringPi

WiringPi on Gordon Hendersonin luoma kirjasto C- ja C++-ohjelmointikielille Raspberry Pin GPIO-tulojen ja lähtöjen käyttämiseksi. WiringPi on asennettava järjestelmään ennen kuin sitä voidaan käyttää. (WiringPi 2015a.)

Asennus tähän järjestelmään tehtiin seuraavasti, lataamalla ensin asennustiedosto */home/pi/robot*-hakemistoon.

Hakemistoon ladattu tiedosto puretaan komennolla:

tar xfz wiringPi-98bcb20.tar.gz

Seuraavaksi siirrytään hakemistoon *cd /home/pi/robot/wiringPi-98bcb20.* Lopuksi vielä käännetään ja asennetaan kirjasto komennolla:

./build

Asennuksen onnistumisen voi tarkistaa seuraavasti:

gpio -v

gpio readall

Edellinen (gpio readall) käsky tulostaa näyttöön listauksen kaikista Raspberry Pi -laitteen tuloista, mikäli asennus on onnistunut. (Wiring Pi 2015b.)

4.1.6 Ohjelmointi- ja testausympäristö

Tässä työssä ohjelmoitua Robologger-ohjelmaa ohjelmoitiin Geany-ohjelmaeditorilla Raspberry Pi -laiteeseen asennettuna. Ohjelman ohjelmointi ja laitteen testaaminen todettiin parhaaksi suorittaa etäyhteyden avulla. Raspberry Pi -tietokoneeseen asennettiin VNC-serveri. Windows-tietokoneessa olevan VNC Viewer -ohjelman avulla saatiin Raspberry Pi -tietokoneen graafinen työpöytä Windows -tietokoneelle. Tällä järjestelyllä Raspberry Pi -tietokoneen käyttämiseksi ei tarvita omaa näyttöä, näppäimistöä ja hiirtä. Sarjaportin testausta voitiin myös suorittaa saman Windows -tietokoneen avulla lähettämällä Windows-koneen sarjaportista aiemmin oikeasta robotista tallennettua näytedataa Raspberry Pi -tietokoneen sarjaporttiin.

4.1.7 Geany-ohjelmaeditorin asetukset

Ohjelmoidun ohjelman kääntäminen Geany-ohjelmaeditorilla voidaan suorittaa F8-näppäimellä. Kääntäminen ja koostaminen ajettavaksi tiedostoksi voidaan suorittaa F9-näppäimellä. Käännetyn ja koostetun ohjelman ajaminen voidaan käynnistää F5-näppäimellä. Näppäinten suorittamat komennot täytyy asetella Set Build Commands -valikossa seuraavasti:

Käännä kenttään seuraava komento:

g++ -Wall -c "%f" -std=c++0x -pthread -lwiringPi

Koosta kenttään seuraava komento:

g++ -Wall -o "%e" "%f" -std=c++0x -pthread -lwiringPi

Suorita kenttään seuraava komento:

sudo "./%e"

4.1.8 Automaattinen kirjautuminen käyttöjärjestelmään

Raspberry Pi on määritelty kirjautumaan automaattisesti käyttöjärjestelmään käynnistyessään. Automaattinen kirjautuminen määritellään */etc/inittab*-tiedostossa. Määritys tapahtuu kommentoimalla seuraava rivi pois #-merkillä:

1:2345:respawn:/sbin/getty –noclear 38400 tty1

Ja lisäämällä seuraava rivi heti edellisen jälkeen.

1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1

4.1.9 C++-ohjelman käynnistys automaattisesti

Raspberry Pin käynnistyessä käyttöjärjestelmään kirjautumisen jälkeen ajettavaan */etc/profile*-tiedostoon on viimeiseksi määritelty ajettavaksi */home/pi/robot/startraspi.sh*-skriptitiedosto. Tämä skriptitiedosto käynnistää ja päivittää tarvittaessa ohjelmoidun Robologger-ohjelman. Skriptin tarkempi toiminta-kuvaus löytyy kohdasta 4.2.7 C++-ohjelman päivittäminen.

4.2 Ohjelman toiminnan kuvaus

Tässä kappaleessa kuvataan ohjelmassa olevia toimintoja perusteluineen.

4.2.1 Sarjaportin alustus

Sarjaportin käsittelemiseksi tässä opinnäytetyössä otettiin käyttöön Linux- ja Unix -käyttöjärjestelmissä toimiva termios.h-kirjasto. Termios.h-kirjasto on ohjelmointirajapinta sarjaportin I/O-käsittelyä varten. Kirjasto otetaan käyttöön #include <termios.h>-määritteellä C++-ohjelman alussa. Sarjaporttia käyttöönotettaessa on tehtävä monia määrityksiä, kuten sarjaportin polku ja nimi, nopeus, luku/kirjoitus ja pariteetti. Tässä opinnäytetyössä määriteltiin sarjaportti vain lukevaksi, koska robotin sarjaporttiin ei ole tarkoitus kirjoittaa mitään. Termios-

kirjastoon perustuvia valmiita määrityksiä on olemassa valmiina muutamia, joista tässä valittiin käyttöön canonical input processing -malli. (Baumann & Frerking 2001).

Canonical input processing -malli sopii parhaiten tässä työssä käytettäväksi ja otettiin käyttöön vain pienin muutoksin. Sarjaportin alustusmäärittely on esitetty liitteessä 2.

Robotti lähettää huoltosarjaporttiin epäsäännöllisin välein erilaisia ja eripituisia rivejä tekstiä päättyen rivinvaihtomerkkiin. Ohjelma alkaa lukea sarjaporttipuskurista dataa, kun puskurissa on vähintään yksi merkki. Ohjelma lukee ja puskuroi luettua dataa aina rivinvaihtomerkkiin asti, jonka jälkeen luettu data siirtyy ohjelmaan jatkokäsittelyä varten. Tämä lukutapa on luonteeltaan blokkaava. Blokkaava luku tarkoittaa, että ohjelma pysähtyy sarjaportin lukukäskyyn, eikä jatka eteenpäin, ennen kuin sarjaporttipuskurissa on dataa. Blokkauksesta johtuen sarjaportin käsittelyä ajetaan omassa säikeessä. Säikeistyksen avulla voidaan sarjaportti toiminnat eriyttää omaksi osaksi, joka ei estä muun ohjelman vapaata kiertoa. Käytetty sarjaportin lukutapa ei myöskään kuluta laitteen prosessoritehoa juuri ollenkaan, ainoastaan lukiessaan rivin ja siirtäessään luetun datan eteenpäin.

4.2.2 Sarjaporttidatan käsittely ja tapahtumarivin muodostus

Sarjaportista saadaan karkeasti ilmaistuna kahdenlaista, selvästi erotettavissa olevaa riviä dataa. Toisen rivin alussa on päiväys ja kellonaika ja toisessa ei ole. Päiväyksen sisältämää tekstiriviä muokataan ja erotellaan omiin sarakkeisiin tulevaa tietokantaa varten. Päiväyksetön tekstirivi tallennetaan muuttamattomana muuttujan s mukana. Alla on esimerkkirivi päiväyksellisestä tekstirivistä.

14-09-01 11:46:25 MC0: type = STATE_CHANGE id = Operational code = 12

Sarjaportin lukema tekstirivi ohjataan string-tyyppiseen muuttujaan s, josta ensin tarkistetaan löytyykö rivin alusta ylimääräisiä merkkejä (S4C+-ohjainyksikössä on), jotka poistetaan. Mahdollisen poiston jälkeen tutkitaan, onko rivin alussa nyt päiväys ja kellonaika. Mikäli päiväys löytyy, se poistetaan, koska robotin oma kello voi

olla asettamatta oikeaan aikaan. Päiväys lisätään myöhemmin Raspberry Pi -laitteen kellosta, kun tapahtumarivi tallennetaan tiedostoon.

Seuraavaksi, jäljelle jääneestä rivistä erotellaan tiedot omiin muuttujiin ja saadaan seuraavat string-tyypin muuttujat: MC, TYPE, ID, CODE. Lopuksi jäljelle jää tyhjä s-muuttuja.

Rivin ollessa nyt erotettuna voidaan muodostaa viisinumeroinen virhekoodi summaamalla CODE-muuttujan sisältö tapahtumaviestin ID-muuttujan sisällön perusteella saatavaan tapahtumanumerosarjaan (katso kohta 2.12 Robotin tapahtumat ja taulukko 2). CODE-muuttuja täytyy ennen summausta muuttaa integer-tyyppiseksi.

Seuraavaksi haetaan laitteen yksilöivä ID-tunniste */home/pi/robot/ID/id.txt*-tiedostosta, joka tallennetaan string-tyyppiseen muuttujaan id. Tässä vaiheessa joudutaan jokaisen tapahtumarivin alkuun tallennettavan järjestysnumeron vuoksi tai jos laite on vasta perustettu tarkistamaan, onko mahdollisesti menossa ohjelman ensimmäinen ohjelmakierros käynnistymisen jälkeen. Jos on, tarkistetaan, onko tallennustiedosto olemassa. Mikäli tallennustiedostoa ei ole, se luodaan ja tällöin järjestysnumerointi alkaa numerosta yksi. Mikäli tallennustiedosto on olemassa ensimmäisen ohjelmakierron aikana, ohjelma hakee tallennustiedoston viimeiseltä riviltä järjestysnumeron pohjaksi viimeiseksi tallennetun järjestysnumeron, josta laskenta jatkuu. Mikäli kyseessä ei ole ohjelman ensimmäinen ohjelmakierros, edelliset tarkistukset ohitetaan ja järjestysnumeroa lasketaan kumuloivasti yhteen, aina ennen tapahtumarivin tallennusta. Tässä vaiheessa vielä tarkistetaan, että s-muuttujassa on sisältöä, eli ei tallenneta turhaan tyhjiä rivejä.

Ennen tallennusta otetaan Raspberry Pi -laitteen päiväys ja kellonaika, jota käytetään tapahtumaviestin aikaleimana. Lisäksi kysytään vielä Linux-järjestelmän käynnissä oloaikaa ja tallennetaan se tapahtumarivin viimeiseen sarakkeeseen tunteina ja minuutteina. Tapahtumarivin tallennus tiedostoon tapahtuu lisäävästi CSV-tiedostomuotoon.

Tallennettava tapahtumarivi tiedostossa on siten muuttujien avulla ilmaistuna seuraavanlainen:

ordernumber;year:month:day;hour-min-sec-
msec;id;virhe;MC;TYPE;ID;CODE;s;uptime;

Tallennettujen tiedostojen sisältöä on esitelty liitteissä 3 - 6.

4.2.3 Säikeistys

Säikeistys on eräänlaista rinnakkain toimivaa moniajoa, jossa käyttöjärjestelmä jakaa suoritettaville säikeille aikasiivuja prosessorille. Aikasiivuja ajetaan prosessorilla vuorotellen, näistä muodostuu näennäinen moniajo. Säikeistyksen käyttöön ottamiseksi täytyy ohjelmassa ottaa käyttöön *pthread.h*-kirjasto, joka ohjaa säikeistyksen toimintaa. Kirjasto otetaan käyttön *#include <pthread.h>*-määritteellä C++-ohjelman alussa. (Barney 2014.)

Säikeistykseen käyttöön liittyy eräitä tarkkaan huomioon otettavia asioita. Eri säikeiden käyttäessä samaa muuttuvaa resurssia, esimerkiksi double-muuttujaa, voi eri säikeiden yhtäaikainen saman resurssin lukeminen ja kirjoittaminen aiheuttaa hallitsemattomia muutoksia. Esimerkiksi jos säie a lukee muuttujaa ja säie b kirjoittaa samaan muuttujaan, voi muuttujan sisältö muuttua lukemisen aikana. Ratkaisu ongelmaan on *mutex.h*-kirjasto, jolla lukitaan muuttuja eri säikeiden käytön ajaksi. Muuttujan ollessa lukittuna toinen säie ei pääse sitä muokkaamaan, ennen kuin toinen säie on purkanut lukituksen. Kirjasto otetaan käyttöön #include <mutex.h> -määritteellä C++-ohjelman alussa. (Barney 2014.)

Taulukossa 5 on esitetty mahdollinen ristiriita summamuuttujan arvossa ilman muuttujan lukitsemista mutexin avulla.

Taulukko 4. Summamuuttuja ilman mutexia (Perustuu Barney 2014).

Säie a	Säie b	Summa
Lue summa: 1000		1000
	Lue summa: 1000	1000
	Lisää summaan: 200	1000
Lisää summaan: 200		1000
Päivitä summa: 1000+200		1200
	Päivitä summa: 1000+200	1200

4.2.4 Käyntiaikalaskuri

Ohjelmaan on ohjelmoitu lisätoimintoina kaksi kappaletta käyntiaikalaskuria. Tuloihin voidaan kytkeä vapaasti eri toimintoja, joiden käyntiaikaa halutaan kulloinkin mitata (kuvio 9). Käyntiaikalaskurit laskevat WiringPi-kirjaston mukaisissa liittimissä 26 ja 27 olevien tulojen tilan perusteella aikaa. Aika otetaan funktiosta, joka generoi ajan millisekunteina. Aika lasketaan kumuloivasti tunneittain tulon HIGH-tilasta. Tulon muuttuessa tilaan HIGH, otetaan ylös aloitusaika. Nyt ohjelma tekee muita toimintojaan normaalisti, kunnes havaitsee tulon muuttuvan tilaan LOW. Tulon muuttuessa tilaan LOW otetaan loppuaika, josta vähennetään aloitusaika. Saatu käyntiaika lisätään itseensä seuraavasti:

käyntiaika = käyntiaika + (loppuaika - aloitusaika)

Lisäksi kumuloitunut käyntiaika tallennetaan tiedostoon tunnin välein. Kumuloitunut käyntiaika nollataan tallennuksen jälkeen. Mikäli tulo on HIGH-tilassa tunnin vaihtumisen aikana, lasketaan käyntiaika yhteen tälle tunnille ja samalla muodostetaan uusi aloitusaika. Tunnin vaihtumisen jälkeen toiminta jatkuu taas normaalisti, laskien käyntiaikaa seuraavalle tunnin ajanjaksolle.

Tallennettaessa käyntiajat näin tunneittain tiedostosta näkee helposti käyntiajat erikseen vuorokauden kaikille tunneille. Tiedostosta on myös helppo muodostaa viivadiagrammi esimerkiksi Excel-taulukkolaskentaohjelmalla havainnollistamaan pitempiaikaista käyntiaikaa tunneittain. Tietokannassa on myös mahdollista helposti laskea käyntiaikoja halutusti.

4.2.5 Kappalelaskuri

Ohjelmaan on ohjelmoitu lisätoimintoina kaksi kappaletta kappalelaskuria. Tuloihin voidaan kytkeä vapaasti eri toimintoja, joiden määrää halutaan kulloinkin laskea (kuvio 9). Kappalelaskurit laskevat WiringPi-kirjaston mukaisissa liittimissä 22 ja 23 olevien tulojen tilan perusteella määrää. Kappalelaskentatuloista on ohjelmallisesti suodatettu kytkinvärähtelyn vaikutus pois.

Käytettäessä tuloja laskentaan on erittäin tärkeää suodattaa mekaanisissa kytkimissä, painonapeissa ja releiden koskettimissa esiintyvä kytkinvärähtely. Ilman suodatusta laskennasta ei saa tarkkaa ohjelman laskiessa kaikki värähtelyt yhteen. Kytkinvärähtelyä esiintyy aina useiden millisekuntien ajan kytkinten vaihtaessa tilaa. Kytkinvärähtelyn aika on riippuvainen kytkinmallista. Kytkinvärähtelyä voi suodattaa ohjelmallisesti tai suodatinpiireillä, kuten esimerkiksi shmitt-trigger- tai RC-piirillä. Ohjelmallisessa suodatuksessa täytyy oskilloskoopilla mitata tai kokeilemalla kokeilla, miten kauan kyseinen kytkintyyppi värähtelee. Ohjelmallisessa suodatuksessa kytkimen nousevasta tai laskevasta reunasta mitataan aikaa, esimerkiksi 20 ms, jos kytkin on vielä tämän ajan jälkeen samansuuntaisessa tilassa, tulkitaan tila vasta vaihtuneeksi. (Ganssle 2008, 1 - 22.)

Tallennettaessa kappalelaskut näin tunneittain tiedostosta näkee helposti kappalemäärät erikseen vuorokauden kaikille tunneille. Tiedostosta on myös helppo muodostaa viivadiagrammi esimerkiksi Excel-taulukkolaskentaohjelmalla havainnollistamaan pitempiaikaista kappalemäärää tunneittain. Tietokannassa on myös mahdollista helposti laskea kappalemääriä halutusti.

4.2.6 Tiedoston tallennus

Ohjelma tallentaa tallennettavaksi määritellyt tiedot kahteen tiedostoon. Ensimmäinen tiedosto sisältää sarjaportin kautta saatavat tapahtumat sekä muut laitteiston tapahtumatiedot. Toinen tiedosto sisältää järjestelmään ohjelmoitujen laskureiden tuntikohtaiset tiedot. Tiedostot nimetään automaattisesti päivänmäärän mukaan seuraavalla kaavalla vuosi_kk_pv_robolog_ID.csv ja vuosi_kk_pv_laskuritlog_ID.csv. Tiedoston nimien alkuun lisätään päivänmäärä, koska tiedostot tallennetaan päiväkohtaisesti. Ohjelma tallentaa tiedostot yöllä edellisen päivänmäärän mukaan nimettynä */home/pi/robot/day_log_files*-hakemistoon. Hakemistossa säilytetään päiväkohtaisia tiedostoja haluttu aika, esimerkiksi kaksi viikkoa. Kaksi viikkoa vanhemmat tiedostot voidaan poistaa tai siirtää muualle Cronin avulla ajastetusti suoritettavalla skriptillä, ettei muistikortti täyty ajan mittaan. Tiedostonimen loppuun yhdistetään myös automaattisesti laitteen ID-numero, joka identifioi tiedoston tallentavaan laitteeseen. Tällä tiedoston nimeämistavalla tiedostot ovat hyvässä järjestyksessä ja laitekohtaisesti nimettynä, jos tiedostoja halutaan siirtää pitempiaikaiseen säilytykseen muualle. Nimeämistapa helpottaa myös määrittelyä vanhempien tiedostojen poistamiseksi.

Järjestelmään luodaan oma ”tietokantaominaisuus” tallentamalla tiedot tietyksi ajaksi, koska laitetta on mahdollista käyttää myös muissa laitteissa tiedonkerääjänä ohjelmoitujen laskentatulojen avulla.

4.2.7 C++-ohjelman päivittäminen

Ohjelmoidun C++-ohjelman päivittäminen voidaan suoritta automaattisesti. Järjestelyssä voidaan Samban ja tiedostonjaon avulla siirtää uusi muutettu, käännetty ja koostettu ohjelmatiedosto */home/pi/robot/UpdateCpp*-hakemistoon. Ohjelmoituun Robologger-ohjelmaan on ohjelmoitu toiminto, joka tarkkailee määritetyn ajan välein, esimerkiksi kerran päivässä, tätä hakemistoa. Ohjelman huomatessa hakemistossa oikean nimisen tiedoston annetaan ohjelmasta *reboot*-komento. Reboot-komento käynnistää Raspberry Pin uudelleen. Raspberry Pin käynnistyessä uudelleen suoritetaan viimeisenä skriptitiedosto *startraspi.sh*. Scriptitiedosto tarkistaa ensin onko uusi päivitystiedosto olemassa, jos on, se kopioidaan */home/pi/robot-*

hakemistoon uudeksi ohjelmatiedostoksi. Kopioinnin jälkeen */home/pi/UpdateCpp*-hakemisto tyhjätään ja käynnistetään ohjelmoitu ohjelma päivitetyllä ohjelmatiedostolla automaattisesti. Lisäksi, mikäli päivityshakemisto on käynnistyksen aikana tyhjä, ohjelma käynnistetään normaalisti olemassa olevalla ohjelmatiedostolla. Päivitystoiminnon on ajastettu toimimaan yöaikaan, kun järjestelmällä ei ole muuta tekemistä. Mikäli päivitys halutaan käyttöön heti, sen voi tehdä paikallisesti tietokoneella tai etäyhteyden avulla.

Tässä päivitysjärjestelyssä oletetaan, että päivitettävä C++-ohjelma testataan, käännetään ja koostetaan valmiiksi vastaavanlaisessa laitekokoonpanossa.

Myöhemmin protovaiheen jälkeen, kun verkkoyhteydet ja muu laitteisto on valmiina, voidaan päivitysskriptiä muokata tarkkailemaan päivitystiedostoa esimerkiksi valvomotietokoneelta. Tällöin päivitetty ohjelmatiedosto on vain yhdessä paikassa, mistä kaikki järjestelmän Raspberry Pi -tietokoneet hakevat päivityksen.

4.2.8 Toiminta sähkökatkossa

Järjestelmälle rakennettu UPS-laite syöttää sähkökatkon aikana laitteelle virtaa maksimissaan yhden minuutin. Sähkökatko tunnistetaan järjestelmässä olevan erillisen sähkökatkoreleen avulla (kuvio 10). Sähkökatkoreleen kärkien kautta on kytketty tieto sähkönsyötön tilasta WiringPi-kirjaston mukaisesti liittimeen 28. Laitteessa oleva Robologger-ohjelma tarkkailee tämän tuloliittimen tilaa. Normaali tila tulolla on ”HIGH” eli verkkojännite kunnossa. Mikäli tuloliittimen tila on yhtäjaksoisesti tilassa ”LOW” yli 30 sekuntia, tulkitaan verkkojännitteen olevan poikki. Tällöin ohjelma antaa Linux-käyttöjärjestelmälle käskyn

sudo shutdown -h now

joka sammuttaa Raspberry Pi -laitteen hallitusti, ennen UPS-kondensaattorien tyhjentymistä. Lyhyemmät alle 30 sekuntia kestävät sähkökatkot eivät vaikuta laitteen toimintaan.

4.3 Uuden laitteen lisääminen

Ensimmäisenä asennetun ja testatun Raspberry Pi -laitteen muistikortista otetaan levykuva (image). Levykuva voidaan ottaa esimerkiksi Win32DiskImager-ohjelmalla. Tämä levykuva toimii varmuuskopiona ja mahdollistaa seuraavien moduulien perustamisen helposti ja nopeasti. Uuden moduulin perustaminen tapahtuu tallentamalla ensimmäisenä määritellystä muistikortista tallennettu levykuva uudelle muistikortille. Tallentamiseen voi käyttää samaa Win32DiskImager-ohjelmaa. Lisäksi uuteen moduulin vaihdetaan uusi IP-osoite ja laitteen yksilöivä ID-numero, joiden on oltava jokaisessa perustettavassa moduulissa yksilölliset. IP-osoitteen ja laitteen yksilöivän ID-numeron määrittely on päätetty suorittaa manuaalisesti. Muuttaminen onnistuu liittämällä moduuli tietokoneeseen verkkokaapelin avulla. Yhteys moduuliin muodostetaan luomalla SSH-yhteys Putty-ohjelmalla, käyttäen alkuperäisessä levykuvassa olevaa IP-osoitetta (192.168.1.30). Yhteyden muodostuttua vaihdetaan staattinen IP-osoite uuteen */etc/network/interfacecs*-tiedostossa ja ID-numero uuteen */home/pi/robot/ID/id.txt*-tiedoston ensimmäiselle riville. Tästä tiedostosta Robologger-ohjelma hakee laitteen yksilöivän ID-numeron tapahtumien yksilöimiseksi moduuliin. Laitteen *hostname* eli nimi verkossa on myös syytä vaihtaa yksilölliseksi. Yksilöllisen nimen avulla voidaan välttyä mahdollisilta virheiltä verkkojaossa, ja laite myös erottuu paremmin verkossa. Hostname vaihdetaan */etc/hosts*-tiedoston viimeisellä rivillä halutuksi. Sarjaportin ja laskureiden datan tallennustiedostot ohjelma luo automaattisesti, mikäli niitä ei ole, joten siitä ei tarvitse huolehtia perustettaessa uutta laitetta. Nyt uusi perustettu moduuli on valmiina asennettavaksi fyysisesti paikoilleen.

5 TULOKSET

Opinnäytetyön tuloksena on saatu toimiva testilaitteisto robotin virhe- ja tilailmoitusten sekä muiden tapahtumailmoitusten lukemiseen ja tallentamiseen (liite 7). Tapahtumien tietojen eriyttäminen onnistui. Lisäksi saatiin muodostettua robotin virheluettelon mukainen virhekoodi eriytetyistä tiedoista, mikä helpottaa viankuvauksien hakua. Tallennettavat tapahtumat tallennetaan csv-tiedostomuotoon. Valittu tiedoston tallennusmuoto mahdollistaa tietojen siirron ja käsittelyn vaivattomasti taulukkolaskenta- ja tietokantaohjelmissa. Ohjelmakoodia on ohjelmoituna reilut 700 riviä kommentteineen.

Käyntiaika- ja kappalelaskenta toiminnat saatiin myös onnistuneesti suoritettua. Molempia toimintoja on ohjelmoituna kaksi kappaletta. Laskentatuloihin voi vapaasti kytkeä haluamiaan tiloja seurattavaksi (katso kappaleet 3.4.1, 4.2.4 ja 4.2.5). Laitetta voi käyttää tarvittaessa laskentatoimintojen osalta myös muissa koneissa ja laitteissa.

Ylimääräisinä toimintoina suunniteltiin kolme lisätoimintoa. Ensimmäisenä oli ohjelmoidun ohjelman automaattinen päivittäminen, mikä helpottaa laitteiston ylläpitoa. Toisena lisätoimintona rakennettiin UPS-varajännitelähde, joka suojaa laitteen tiedostojärjestelmää sähkökatkojen varalta. UPS-varajännitelähde mahdollistaa myös robotin sammutuksen aikaisten tapahtumien tallentamisen sähkökatkon aikana. Kolmantena laiteelle rakennettiin myös protopiirilevy, jossa on riviliittimet tulojen ja lähtöjen liittämiseksi, sekä toimintalogiikka laitteen uudelleen käynnistämiseksi, automaattisesti sähkökatkon jälkeen.

6 POHDINTA JA YHTEENVETO

Opinnäytetyö on ollut tekijälleen mielenkiintoinen ja opettavainen sulautettujen järjestelmien projekti. Työssä konkretisoituu erittäin hyvin monet sulautettujen järjestelmien osa-alueet, joiden hallinta auttaa jatkossa vastaavanlaisten järjestelmien suunnittelua ja valmistusta. Työ on herättänyt jonkin verran ajatuksia ja kysymyksiä, mistä koulussa ei ole ollut puhetta. Toisaalta koulun opeista on ollut monessa kohtaa hyötyäkin. Tiedon hakuun on käytetty paljon aikaa ja siinä sivussa on luettu paljon aiheesta ja aiheen sivustakin erilaista tietoa. Opinnäytetyö saatiin kuitenkin päätökseen suunnitellussa ajassa. Opinnäytetyötä on tehty osaksi työn ohessa vapaa-aikana, sekä tekijän lyhyen opintovapaan aikana. Ilman pidettyä opintovapaata opinnäytetyö ei ehkä olisi valmistunut ajallaan tai tekijää miellyttävällä tavalla. Testaustoimintoja on tehty töissä oikeassa toimintaympäristössä. Työn alkuvaiheessa kerättiin näytedataa kahdesta käytössä olevasta kontrollerityypistä. Loppuvaiheessa laite asennettiin paikoilleen ja testattiin toiminnat myös oikeassa toimintaympäristössä. Tehty ohjelma osasi lukea ja tallentaa molempien ohjainyksikkötyyppien (IRC 5 ja S4C+) tuottamat tapahtumat.

Järjestelmän toiminnan suunnittelu ja määrittely sujui ongelmitta. Ajatukset järjestelmältä vaadittavilta ominaisuuksilta tilaavan yrityksen kanssa olivat hyvin yhteneväiset. Käytettävät laitteet saatiin valittua ja tilattua nopeasti, kun järjestelmältä vaadittavat ominaisuudet saatiin määriteltyä. Määrittelyvaiheessa suurin ihmetyksen aihe oli kuinka paljon ja minkälaista dataa sarjaportista tulee. Mitään dokumentointia robotin sarjaportista ei tässä vaiheessa ollut tiedossa. Huoltosarjaportista saatava tieto ja sen muoto selvisi kuitenkin kytkemällä kannettava tietokone sarjaporttiin ja lukemalla terminaaliohjelmalla sarjaporttia.

Seuraavaksi alkoi ohjelmointityö ja sen suunnittelu käyttöjärjestelmän asennuksen ja asetusten määrittelyn jälkeen. Ohjelmointityö oli kirjoittajalle henkilökohtaisesti projektin suurin mielenkiinnon ja samalla myös ahdistuksen aihe, koska kokemusta C++-kielen ohjelmoinnista ei juuri ollut. Paljon työtä, testausta ja C++-kielen kirjallisuutta ja internetlähteitä tutkimalla on kuitenkin päästy ohjelmoimaan järjestelmän toiminta määrittelyjen mukaiseksi. Suurin yksittäinen ongelma ja aikaa vaativin prosessi ohjelmointityössä oli sarjaportin käyttöönotto. Oikeanlaisen lukumallin

ja asetusten löytäminen vei paljon aikaa. Sarjaportin saaminen toimintaan olikin ohjelmointityön ensimmäinen tavoite, koska muu ohjelmointi perustuu suurimmalta osalta sarjaportista saatavan datan käsittelyyn. Automaattisen päivitystoiminnon miettiminen ja suunnittelu oli ajatuksissa muun ohjelmoinnin rinnalla pitkän aikaa. Päivitystoiminnasta ei löytynyt mitään mallia tai lähdettä, jonka pohjalta toimintaa olisi voinut alkaa toteuttaa. Tässä aika teki tehtävänsä ja päivitystoiminnon käyttöönotto sujui lopulta nopeasti, kun ajatus toteutustavasta selvisi. Päivitystoiminto onnistui mielestäni hyvin ja onkin lopulta yksinkertainen ja säädeltävissä oleva toiminto (katso kappale 4.2.7).

Opinnäytetyön edetessä tärkeäksi kehityskohteeksi tuli UPS-laitteen tarve. Aikaisemmasta kokemuksesta ja lähteitä lukemalla todettiin, että Raspberry Pi -tietokoneen tiedostojärjestelmän on mahdollista vioittua äkillisissä jännitekatkoksissa. Tämäntyyppisessä teollisessa sovelluksessa on erittäin tärkeää järjestää laitteelle katkeamaton jännitteensyöttö huoltotoimintojen minimoimiseksi. UPS-laitteeksi haluttiin huoltovapaa laite, joten akkuihin ja paristoihin tukeutuvat laitteet eivät tulleet kyseeseen. Päätettiin valita kondensaattoreihin perustuva UPS-laite. Muutamien erilaisten toteutusmallien pohjalta päädyttiin toteuttamaan kuviossa 10 esitetyn mukainen UPS-varajännitelähde. Toteutettu UPS-varajännitelähde onkin toiminnaltaan yksinkertainen ja huoltovapaa laite. Tässä kuitenkin ilmeni ongelmia Raspberry Pi -tietokoneen automaattisen käynnistymisen osalta sähkökatkon jälkeen. Ongelman ratkaisemiseksi kehiteltiin kaksi erilaista ratkaisumallia. Ensimmäisessä Raspberry Pi on sammutettava aina sähkökatkon aikana. Toisessa tavassa Raspberry Pi on sammutettava vasta, jos sähkökatko kestää yhtäjaksoisesti ohjelmassa määriteltyä aikaa kauemmin. Tässä työssä testattiin ja otettiin käyttöön toisena mainittu vaihtoehto, joka toimikin oikein hyvin (katso kappaleet 2.11 ja 3.7).

Opinnäytetyön aikana pienenä yllätyksenä ilmeni myös Raspberry Pi -tietokoneessa käytettävän muistikortin rajallinen käyttöikä P/E-sykleillä mitattuna. Muistikortin laatuun ja muistin kokoon on kiinnitettävä huomiota. Muistikortissa käytettävän Flash-teknologian, sekä tallennuksen hallinnan valinta on merkityksellistä, varsinkin paljon tallentavissa sovelluksissa. Muistikortin ominaisuuksien selvittämiseen tulikin käytettyä myös paljon aikaa (katso kappale 2.10).

Jatkokehityksen osalta järjestelmään on mahdollista rakentaa uusia toimintoja, sekä kehittää lisää olemassa olevia toimintoja. Yksi jatkokehityskohde on esimerkiksi liikennöintiprotokolla tietojen siirtoon valvomokoneelle. Reaaliaikakellon käyttöönottoa voi myös pohtia. Reaaliaikakello mahdollistaa laitteen pysymisen oikeassa ajassa mahdollisten verkkokatkosten tai valvomotietokoneen ongelmien aikana. Jatkossa tietokannan puolella voitaisiin haluttaessa muodostetun virhenumeron perusteella tuoda näkyville robotin käyttäjänoppaan vianmääritysosion mukaiset tapahtumakuvaustekstit. Tämä toiminto yksinkertaistaisi ja nopeuttaisi tapahtumien ymmärtämistä, sekä nopeuttaisi vian hakua vielä lisää.

Yhteenvetona voin olla tyytyväinen onnistuneesti rakennettuun järjestelmään. Oppia ja tietotaitoa on tullut tekijälle. Tehty järjestelmä täyttää määritellyt tavoitteet ja enemmänkin. Samalla olen oppinut C++-kielen ohjelmointia, jonka taidon kehittämisen olen asettanut tärkeäksi tavoitteeksi valitessani opinnäytetyön aihetta. Joskin ohjelmointikielen laajuuden vuoksi, C++-kielen ohjelmoinnin saralla kuljetaan vielä pienin askelin, mutta eteenpäin kuitenkin. Tässä voin todeta ja allekirjoittaa monissa lukemissani lähdeteoksissa mainitun sanonnan ”ohjelmointia oppii ohjelmoimalla” huomatessani ohjelmointitaitojeni kehittyvän.

LÄHTEET

ABB. 2008a. Controller IRC 5 with FlexPendant. [DVD-levy]. ABB ab. [Viitattu 9.3.2015]. Saatavissa: Documentation for Robotics Products M2004/IRC 5.

ABB. 2008b. Vianmääritysopas, IRC 5. [DVD-levy]. ABB ab. [Viitattu 9.3.2015]. Saatavissa: Documentation for Robotics Products M2004/IRC 5.

ABB. 2012. IRB 340. [pdf-dokumentti]. ABB Oy. [Viitattu 10.11.2014]. Saatavissa: http://www04.abb.com/global/seitp/seitp202.nsf/c71c66c1f02e6575c125711f004660e6/d3900b3c7489746bc12571b700351d6c/$FILE/Data+Sheet+IRB+340HR.pdf

ABB. Ei päiväystä. ABB Suomessa. [www-dokumentti]. ABB Oy. [Viitattu 10.12.2014]. Saatavissa: http://www.abb.fi/product/seitp327/f34aac99b4a5f32bc125749c002cc7e8.aspx?productLanguage=fi&country=FI&tabKey=2

ABB. Ei päiväystä. Robotics. [pdf-dokumentti]. ABB Oy. [Viitattu 8.11.2014]. Saatavissa: http://www05.abb.com/global/scot/scot241.nsf/veritydisplay/98ba43a906331fec48257c6f00374818/$file/pr10031en%20r15_en.pdf

AB Electronics UK. Ei päiväystä. Serial Pi. [pdf-dokumentti]. Apexweb Ltd. [Viitattu 29.9.2014]. Saatavissa: https://www.abelectronics.co.uk/docs/stock/raspberrypi/serialpi/Datasheet-SerialPi.pdf

AB Electronics UK. Ei päiväystä. Using the Serial Port. [www-dokumentti]. Apexweb Ltd. [Viitattu 10.3.2015]. Saatavissa: https://www.abelectronics.co.uk/raspberrypi-serialportusage/info.aspx

Atria. Ei päiväystä. Atria-konserni. [www-lähde]. Atria Oyj. [Viitattu 1.11.2014]. Saatavissa: http://www.atriagroup.com/atria-konserni/Sivut/default.aspx

Barney, B., 2014. POSIX Threads Programming. [www-dokumentti]. Lawrence Livermore National Laboratory. [Viitattu 10.3.2015]. Saatavissa: https://computing.llnl.gov/tutorials/pthreads/

Baumann, P. & Frerking, G., 2001. Serial Programming HOWTO. [www-dokumentti]. www.tldp.org. [Viitattu 10.3.2015]. Saatavissa: http://www.tldp.org/HOWTO/Serial-Programming-HOWTO/x115.html#AEN125

Freeman, C. 2014. What are the best SD cards to use in a Raspberry Pi?. [WWW-dokumentti]. RepRage.com. [Viitattu 4.2.2015]. Saatavissa: http://reprage.com/post/what-are-the-best-sd-cards-to-use-in-a-raspberry-pi/

Ganssle, J., 2008. A Guide to Debouncing. [pdf-dokumentti]. The Ganssle Group. [Viitattu 10.3.2015]. Saatavissa: http://www.eng.utah.edu/~cs5780/debouncing.pdf

Hovi, A., Huotari, J. & Lahdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

Kingston. Ei päiväystä. Flash memory guide. [pdf-dokumentti]. Kingston Techonology Corporation. [Viitattu 4.2.2015]. Saatavissa: http://media.kingston.com/pdfs/FlashMemGuide.pdf

Kuivanen, R (toim.). 1999. Robotiikka. Vantaa: Talentum Oyj / Metallitekniikka.

Lehtinen, H. Ei päiväystä. Robotit. [www-lähde]. Suomen Automaatioseura ry. [Viitattu 8.11.2014]. Saatavissa: http://www.google.fi/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=1&ved=0CCcQFjAA&url=http%3A%2F%2Fwww.automaatioseura.fi%2Findex%2Ftiedos-tot%2FRobotit.doc&ei=axZeVMSWOtLgaomFgKgC&usg=AFQjCNHLbNrWfnXmCIYuOdxELCT7PIb6Kg

Oracle. Ei päiväystä. MySQL Community Downloads. [WWW-dokumentti]. Oracle Corporation. [Viitattu 12.2.2015]. Saatavissa: http://www.oracle.com/us/products/mysql/overview/index.html

Raspbian. Ei päiväystä. Welcome to Raspbian. [www-dokumentti]. Raspbian.org. [Viitattu 29.9.2014]. Saatavissa: http://www.raspbian.org/

Raspberry Pi Foundation. Ei päiväystä. Faqs. [www-dokumentti]. Raspberry Pi Foundation. [Viitattu 11.11.2014]. Saatavissa: http://www.raspberrypi.org/help/faqs/

Raspberry Pi Foundation. Ei päiväystä. NOOBS setup. [www-dokumentti]. Raspberry Pi Foundation. [Viitattu 10.3.2015]. Saatavissa: http://www.raspberrypi.org/help/noobs-setup/

Raspberry Pi Foundation. Ei päiväystä. What is Raspberry Pi. [www-dokumentti]. Raspberry Pi Foundation. [Viitattu 10.11.2014]. Saatavissa: http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

Raspberry Pi. Ei päiväystä. Model B+. [pdf-dokumentti]. Adafruit. [Viitattu 15.10.2014]. Saatavissa: https://www.adafruit.com/datasheets/pi-specs.pdf

SiliconSystems. 2005. Increasing Flash SSD Realibility. [WWW-dokumentti]. Silicon Systems Inc. [Viitattu 5.2.2015]. Saatavissa: http://www.storagesearch.com/siliconsys-art1.html

Stackexchange. 2012. How can i extend the life of my SD card?. [WWW-dokumentti]. Stackexchange.com. [Viitattu 5.2.2015]. Saatavissa: http://raspberrypi.stackexchange.com/questions/169/how-can-i-extend-the-life-of-my-sd-card

Stroustrub, B. 2000. C++-ohjelmointi. Suomentaja Ketola, V-P. Jyväskylä: Teknolit Oy.

Tröger, E. Brush, M. Wendling, C. Laniz, F. Treleaven, N. & Hopf, T. 2010. About Geany. [WWW-dokumentti]. Tröger Enrico. Brush Matthew. Wendling Colomban. Laniz Frank. Treleaven Nick, Hopf, Dominic.[Viitattu 13.2.2015]. Saatavissa: http://www.geany.org/Main/About

Wiring Pi. 2015a. GPIO interface library for the Raspberry Pi. [WWW-dokumentti]. Wiring Pi. [Viitattu 13.2.2015]. Saatavissa: http://wiringpi.com/

Wiring Pi. 2015b. GPIO interface library for the Raspberry Pi. [WWW-dokumentti]. Wiring Pi. [Viitattu 13.2.2015]. Saatavissa: http://wiringpi.com/download-and-install/

LIITTEET

LIITE 1: Raspberry Pi, automaattisen käynnistyksen ajastuksen piirikuva

LIITE 2: Sarjaportin alustusmäärittelyt

LIITE 3: Robolog-tiedoston sisältöä

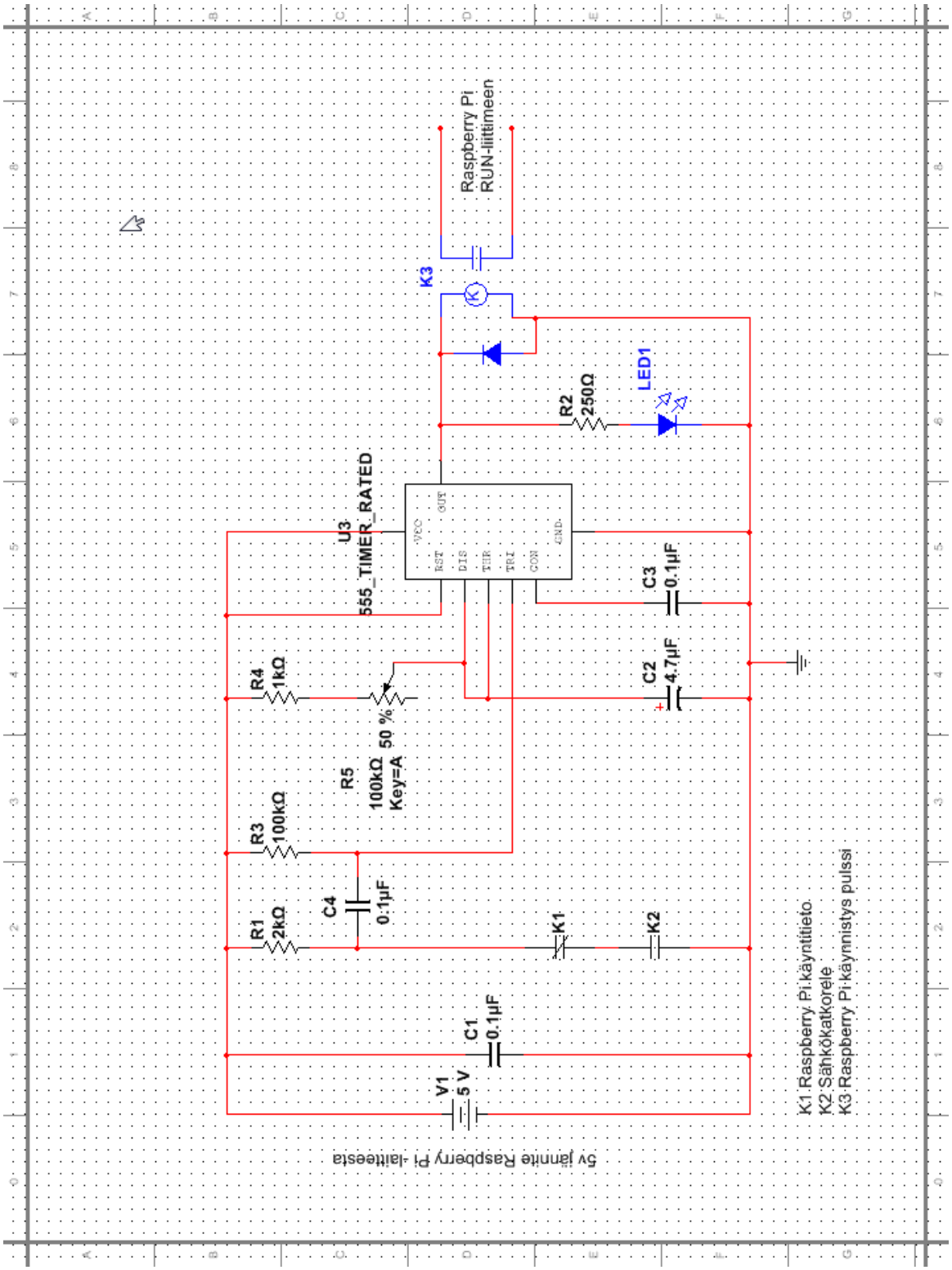
LIITE 4: Robolog-tiedosto Excel-taulukkolaskentaohjelmassa

LIITE 5: Laskuritlog-tiedoston sisältöä

LIITE 6: Laskuritlog-tiedosto Excel-taulukkolaskentaohjelmassa

LIITE 7: Valokuva rakennetusta laitteesta

Liite 1: Raspberry Pi, automaattisen käynnistyksen ajastuksen piirikuva



Liite 2: Sarjaportin alustusmäärittelyt

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>

using namespace std;

/* baudrate settings are defined in <asm/termbits.h>, which
is included by <termios.h> */
#define BAUDRATE B9600
/* change this definition for the correct port */
#define MODEMDEVICE "/dev/ttyAMA0"
#define _POSIX_SOURCE 1 /* POSIX compliant source */
#define FALSE 0
#define TRUE 1

volatile int STOP=FALSE;

main()

{
int fd, res;
struct termios oldtio,newtio;
char buf[255];

/*
  Open modem device for reading and writing and not as controlling tty
  because we don't want to get killed if linenoise sends CTRL-C.
*/

fd = open(MODEMDEVICE, O_RDONLY | O_NOCTTY);
if (fd <0) {perror(MODEMDEVICE); exit(-1); }

tcgetattr(fd,&oldtio); /* save current serial port settings */
bzero(&newtio, sizeof(newtio)); /* clear struct for new port settings */

/*
BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
CRTSCTS : output hardware flow control (only used if the cable has
            all necessary lines. See sect. 7 of Serial-HOWTO)
CS8     : 8n1 (8bit,no parity,1 stopbit)
CLOCAL  : local connection, no modem contol
CREAD   : enable receiving characters
*/
newtio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
/*
IGNPAR  : ignore bytes with parity errors
ICRNL   : map CR to NL (otherwise a CR input on the other computer
            will not terminate input)
        otherwise make device raw (no other input processing)
*/
newtio.c_iflag = IGNPAR | ICRNL;

/*
Raw output.
*/
newtio.c_oflag = 0;
```

```
/*
ICANON  : enable canonical input
disable all echo functionality, and don't send signals to calling program
*/
newtio.c_lflag = ICANON;

/*
initialize all control characters
default values can be found in /usr/include/termios.h, and are given
in the comments, but we don't need them here
*/
newtio.c_cc[VINTR]    = 0;   /* Ctrl-c */
newtio.c_cc[VQUIT]    = 0;   /* Ctrl-\ */
newtio.c_cc[VERASE]   = 0;   /* del */
newtio.c_cc[VKILL]    = 0;   /* @ */
newtio.c_cc[VEOF]     = 4;   /* Ctrl-d */
newtio.c_cc[VTIME]    = 0;   /* inter-character timer unused */
newtio.c_cc[VMIN]     = 1;   /* blocking read until 1 character arrives*/
newtio.c_cc[VSWTC]    = 0;   /* '\0' */
newtio.c_cc[VSTART]   = 0;   /* Ctrl-q */
newtio.c_cc[VSTOP]    = 0;   /* Ctrl-s */
newtio.c_cc[VSUSP]    = 0;   /* Ctrl-z */
newtio.c_cc[VEOL]     = 0;   /* '\0' */

// now clean the modem line and activate the settings for the port

tcflush(fd, TCIFLUSH);
tcsetattr(fd,TCSANOW,&newtio);

/*
  terminal settings done, now handle input
  In this example, inputting a 'z' at the beginning of a line will
  exit the program.
*/

while (STOP==FALSE) /* loop until we have a terminating condition */
{

    // Koodia


}//while (STOP==FALSE)

/* restore the old port settings */
tcsetattr(fd,TCSANOW,&oldtio);
} //main
```

Liite 3: Robolog-tiedoston sisältöä

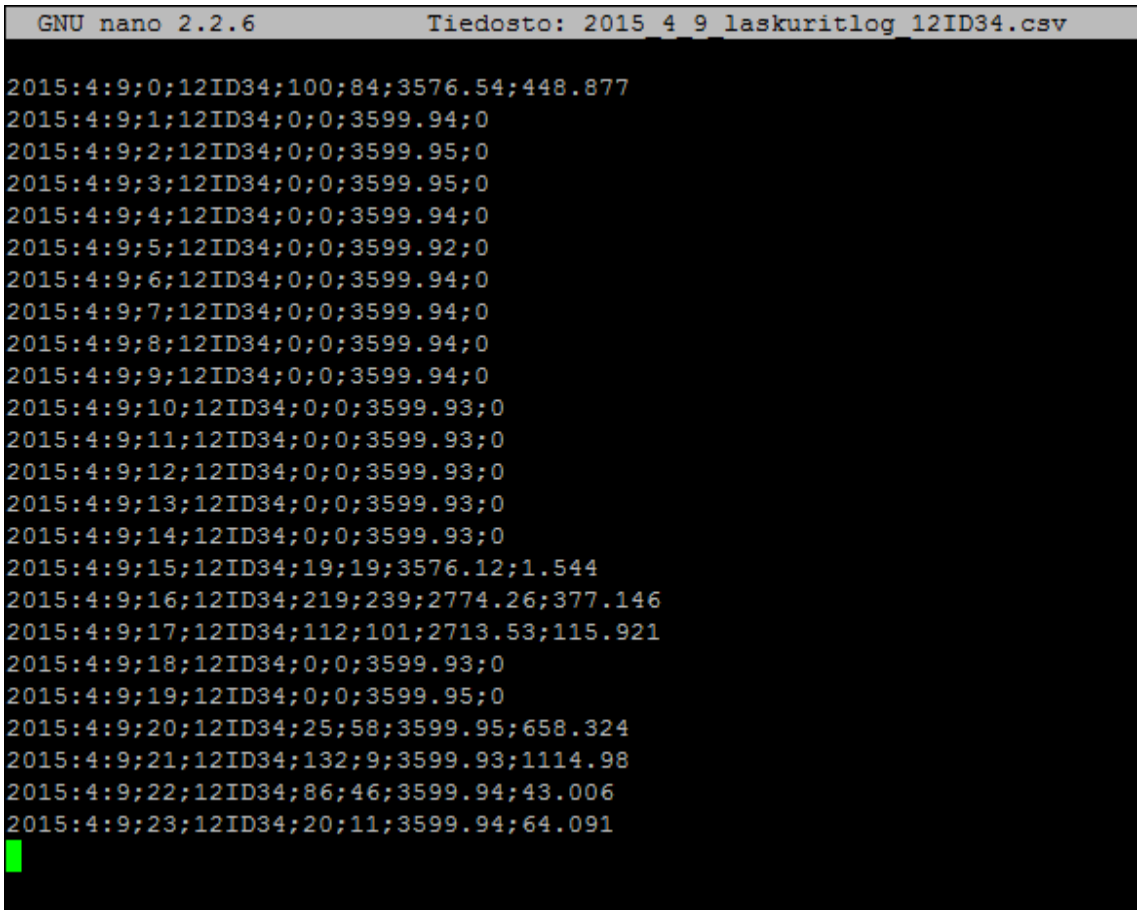
```
 GNU nano 2.2.6       Tiedosto: 2015_4_10_robolog_12ID34.csv


1;2015:4:10;9-25-39-938;12ID34;0;;;;;7231971:267778 : Item source deleted, name: $
2;2015:4:10;9-25-40-203;12ID34;0;;;;;7231971:533243 : Item source deleted, name: $
3;2015:4:10;9-25-40-278;12ID34;10125;MC0;STATE_CHANGE;OPERATIONAL;125; ;0h.3min;
4;2015:4:10;9-25-40-295;12ID34;0;;;;;arg 0: 1 ;0h.3min;
5;2015:4:10;9-25-40-310;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
6;2015:4:10;9-25-45-463;12ID34;10002;MC0;STATE_CHANGE;OPERATIONAL;2; ;0h.3min;
7;2015:4:10;9-25-45-488;12ID34;0;;;;;arg 0: 1 arg 1: 0 ;0h.3min;
8;2015:4:10;9-25-45-504;12ID34;0;;;;;arg 2: T_ROB1;0h.3min;
9;2015:4:10;9-25-45-579;12ID34;10150;MC0;STATE_CHANGE;OPERATIONAL;150; ;0h.3min;
10;2015:4:10;9-25-45-596;12ID34;0;;;;;arg 0: 1 ;0h.3min;
11;2015:4:10;9-25-45-612;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
12;2015:4:10;9-25-45-687;12ID34;10002;MC0;STATE_CHANGE;OPERATIONAL;2; ;0h.3min;
13;2015:4:10;9-25-45-712;12ID34;0;;;;;arg 0: 1 arg 1: 0 ;0h.3min;
14;2015:4:10;9-25-45-721;12ID34;0;;;;;arg 2: T_ROB1;0h.3min;
15;2015:4:10;9-25-45-804;12ID34;10002;MC0;STATE_CHANGE;OPERATIONAL;2; ;0h.3min;
16;2015:4:10;9-25-45-820;12ID34;0;;;;;arg 0: 1 arg 1: 0 ;0h.3min;
17;2015:4:10;9-25-45-837;12ID34;0;;;;;arg 2: T_ROB1;0h.3min;
18;2015:4:10;9-25-45-920;12ID34;10129;MC0;STATE_CHANGE;OPERATIONAL;129; ;0h.3min;
19;2015:4:10;9-25-45-929;12ID34;0;;;;;arg 0: 1 ;0h.3min;
20;2015:4:10;9-25-45-945;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
21;2015:4:10;9-25-46-732;12ID34;10150;MC0;STATE_CHANGE;OPERATIONAL;150; ;0h.3min;
22;2015:4:10;9-25-46-741;12ID34;0;;;;;arg 0: 1 ;0h.3min;
23;2015:4:10;9-25-46-757;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
24;2015:4:10;9-25-46-841;12ID34;10129;MC0;STATE_CHANGE;OPERATIONAL;129; ;0h.3min;
25;2015:4:10;9-25-46-849;12ID34;0;;;;;arg 0: 1 ;0h.3min;
26;2015:4:10;9-25-46-867;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
27;2015:4:10;9-25-46-978;12ID34;10052;MC0;STATE_CHANGE;OPERATIONAL;52; ;0h.3min;
28;2015:4:10;9-25-46-986;12ID34;0;;;;;arg 0: 0 ;0h.3min;
29;2015:4:10;9-25-47-61;12ID34;10053;MC0;STATE_CHANGE;OPERATIONAL;53; ;0h.3min;
30;2015:4:10;9-25-47-78;12ID34;0;;;;;arg 0: 0 ;0h.3min;
31;2015:4:10;9-25-47-153;12ID34;10151;MC0;STATE_CHANGE;OPERATIONAL;151; ;0h.3min;
32;2015:4:10;9-25-47-161;12ID34;0;;;;;arg 0: 1 ;0h.3min;
33;2015:4:10;9-25-47-177;12ID34;0;;;;;arg 1: T_ROB1;0h.3min;
34;2015:4:10;9-25-47-357;12ID34;0;;;;;7231978:688541 : Item source created, name:$
35;2015:4:10;9-25-47-872;12ID34;0;;;;;7231979:204554 : Item source created, name:$
36;2015:4:10;9-27-5-987;12ID34;0;;;;;(safevtts): SYS STOP - from runchn.c,3322;0h$
37;2015:4:10;9-27-6-65;12ID34;10012;MC0;STATE_CHANGE;OPERATIONAL;12; ;0h.5min;
38;2015:4:10;9-27-6-505;12ID34;10015;MC0;STATE_CHANGE;OPERATIONAL;15; ;0h.5min;
39;2015:4:10;9-27-8-499;12ID34;10125;MC0;STATE_CHANGE;OPERATIONAL;125; ;0h.5min;
40;2015:4:10;9-27-8-507;12ID34;0;;;;;arg 0: 1 ;0h.5min;
41;2015:4:10;9-27-8-526;12ID34;0;;;;;arg 1: T_ROB1;0h.5min;
42;2015:4:10;9-27-10-331;12ID34;10016;MC0;STATE_CHANGE;OPERATIONAL;16; ;0h.5min;
43;2015:4:10;9-27-10-479;12ID34;10010;MC0;STATE_CHANGE;OPERATIONAL;10; ;0h.5min;
44;2015:4:10;9-27-11-709;12ID34;40210;MC0;WARNING;PROGRAM;210; ;0h.5min;
45;2015:4:10;9-27-11-726;12ID34;0;;;;;arg 0: T_ROB1;0h.5min;
46;2015:4:10;9-27-14-296;12ID34;10017;MC0;STATE_CHANGE;OPERATIONAL;17; ;0h.5min;
47;2015:4:10;9-27-24-742;12ID34;10011;MC0;STATE_CHANGE;OPERATIONAL;11; ;0h.5min;
48;2015:4:10;9-27-24-952;12ID34;10052;MC0;STATE_CHANGE;OPERATIONAL;52; ;0h.5min;
49;2015:4:10;9-27-24-960;12ID34;0;;;;;arg 0: 0 ;0h.5min;
50;2015:4:10;9-27-25-35;12ID34;10053;MC0;STATE_CHANGE;OPERATIONAL;53; ;0h.5min;
51;2015:4:10;9-27-25-51;12ID34;0;;;;;arg 0: 0 ;0h.5min;
52;2015:4:10;9-27-25-127;12ID34;10156;MC0;STATE_CHANGE;OPERATIONAL;156; ;0h.5min;
```

Liite 4: Robolog-tiedosto Excel-taulukkolaskentaohjelmassa

Järjnro	Päiväys	Aika	id	Virhe nro	MC	TYPE	ID	CODE	MESSAGE	Linux uptime
1	2015:04:10	9-25-39-938	12ID34	0					7231971:267778 : Item source deleted, name: ItmSrcCnv1, id: 27, mem	0h.3min
2	2015:04:10	9-25-40-203	12ID34	0					7231971:533243 : Item source deleted, name: ItmSrcCnv2, id: 26, mem	0h.3min
3	2015:04:10	9-25-40-278	12ID34	10125	MC0	STATE_CHANGE	OPERATIONAL	125		0h.3min
4	2015:04:10	9-25-40-295	12ID34	0					arg 0: 1	0h.3min
5	2015:04:10	9-25-40-310	12ID34	0					arg 1: T_ROB1	0h.3min
6	2015:04:10	9-25-45-463	12ID34	10002	MC0	STATE_CHANGE	OPERATIONAL	2		0h.3min
7	2015:04:10	9-25-45-488	12ID34	0					arg 0: 1 arg 1: 0	0h.3min
8	2015:04:10	9-25-45-504	12ID34	0					arg 2: T_ROB1	0h.3min
9	2015:04:10	9-25-45-579	12ID34	10150	MC0	STATE_CHANGE	OPERATIONAL	150		0h.3min
10	2015:04:10	9-25-45-596	12ID34	0					arg 0: 1	0h.3min
11	2015:04:10	9-25-45-612	12ID34	0					arg 1: T_ROB1	0h.3min
12	2015:04:10	9-25-45-687	12ID34	10002	MC0	STATE_CHANGE	OPERATIONAL	2		0h.3min
13	2015:04:10	9-25-45-712	12ID34	0					arg 0: 1 arg 1: 0	0h.3min
14	2015:04:10	9-25-45-721	12ID34	0					arg 2: T_ROB1	0h.3min
15	2015:04:10	9-25-45-804	12ID34	10002	MC0	STATE_CHANGE	OPERATIONAL	2		0h.3min
16	2015:04:10	9-25-45-820	12ID34	0					arg 0: 1 arg 1: 0	0h.3min
17	2015:04:10	9-25-45-837	12ID34	0					arg 2: T_ROB1	0h.3min
18	2015:04:10	9-25-45-920	12ID34	10129	MC0	STATE_CHANGE	OPERATIONAL	129		0h.3min
19	2015:04:10	9-25-45-929	12ID34	0					arg 0: 1	0h.3min
20	2015:04:10	9-25-45-945	12ID34	0					arg 1: T_ROB1	0h.3min
21	2015:04:10	9-25-46-732	12ID34	10150	MC0	STATE_CHANGE	OPERATIONAL	150		0h.3min
22	2015:04:10	9-25-46-741	12ID34	0					arg 0: 1	0h.3min
23	2015:04:10	9-25-46-757	12ID34	0					arg 1: T_ROB1	0h.3min
24	2015:04:10	9-25-46-841	12ID34	10129	MC0	STATE_CHANGE	OPERATIONAL	129		0h.3min
25	2015:04:10	9-25-46-849	12ID34	0					arg 0: 1	0h.3min
26	2015:04:10	9-25-46-867	12ID34	0					arg 1: T_ROB1	0h.3min
27	2015:04:10	9-25-46-978	12ID34	10052	MC0	STATE_CHANGE	OPERATIONAL	52		0h.3min
28	2015:04:10	9-25-46-986	12ID34	0					arg 0: 0	0h.3min
29	2015:04:10	9-25-47-61	12ID34	10053	MC0	STATE_CHANGE	OPERATIONAL	53		0h.3min
30	2015:04:10	9-25-47-78	12ID34	0					arg 0: 0	0h.3min
31	2015:04:10	9-25-47-153	12ID34	10151	MC0	STATE_CHANGE	OPERATIONAL	151		0h.3min
32	2015:04:10	9-25-47-161	12ID34	0					arg 0: 1	0h.3min
33	2015:04:10	9-25-47-177	12ID34	0					arg 1: T_ROB1	0h.3min
34	2015:04:10	9-25-47-357	12ID34	0					7231978:688541 : Item source created, name: ItmSrcCnv1, id: 22, mem	0h.3min
35	2015:04:10	9-25-47-872	12ID34	0					7231979:204554 : Item source created, name: ItmSrcCnv2, id: 21, mem	0h.3min
36	2015:04:10	9-27-5-987	12ID34	0					(safevtts): SYS STOP - from runchn.c,3322	0h.5min
37	2015:04:10	9-27-6-65	12ID34	10012	MC0	STATE_CHANGE	OPERATIONAL	12		0h.5min
38	2015:04:10	9-27-6-505	12ID34	10015	MC0	STATE_CHANGE	OPERATIONAL	15		0h.5min
39	2015:04:10	9-27-8-499	12ID34	10125	MC0	STATE_CHANGE	OPERATIONAL	125		0h.5min
40	2015:04:10	9-27-8-507	12ID34	0					arg 0: 1	0h.5min
41	2015:04:10	9-27-8-526	12ID34	0					arg 1: T_ROB1	0h.5min
42	2015:04:10	9-27-10-331	12ID34	10016	MC0	STATE_CHANGE	OPERATIONAL	16		0h.5min
43	2015:04:10	9-27-10-479	12ID34	10010	MC0	STATE_CHANGE	OPERATIONAL	10		0h.5min
44	2015:04:10	9-27-11-709	12ID34	40210	MC0	WARNING	PROGRAM	210		0h.5min
45	2015:04:10	9-27-11-726	12ID34	0					arg 0: T_ROB1	0h.5min
46	2015:04:10	9-27-14-296	12ID34	10017	MC0	STATE_CHANGE	OPERATIONAL	17		0h.5min
47	2015:04:10	9-27-24-742	12ID34	10011	MC0	STATE_CHANGE	OPERATIONAL	11		0h.5min
48	2015:04:10	9-27-24-952	12ID34	10052	MC0	STATE_CHANGE	OPERATIONAL	52		0h.5min
49	2015:04:10	9-27-24-960	12ID34	0					arg 0: 0	0h.5min
50	2015:04:10	9-27-25-35	12ID34	10053	MC0	STATE_CHANGE	OPERATIONAL	53		0h.5min

Liite 5: Laskuritlog-tiedoston sisältöä



Liite 6: Laskuritlog-tiedosto Excel-taulukkolaskentaohjelmassa

Päiväys	Tunti	ID	Kpl_1	Kpl_2	Käy_1 / sek	Käy_2 / sek
2015:04:09	0	12ID34	100	84	3576.54	448.877
2015:04:09	1	12ID34	0	0	3599.94	0
2015:04:09	2	12ID34	0	0	3599.95	0
2015:04:09	3	12ID34	0	0	3599.95	0
2015:04:09	4	12ID34	0	0	3599.94	0
2015:04:09	5	12ID34	0	0	3599.92	0
2015:04:09	6	12ID34	0	0	3599.94	0
2015:04:09	7	12ID34	0	0	3599.94	0
2015:04:09	8	12ID34	0	0	3599.94	0
2015:04:09	9	12ID34	0	0	3599.94	0
2015:04:09	10	12ID34	0	0	3599.93	0
2015:04:09	11	12ID34	0	0	3599.93	0
2015:04:09	12	12ID34	0	0	3599.93	0
2015:04:09	13	12ID34	0	0	3599.93	0
2015:04:09	14	12ID34	0	0	3599.93	0
2015:04:09	15	12ID34	19	19	3576.12	1.544
2015:04:09	16	12ID34	219	239	2774.26	377.146
2015:04:09	17	12ID34	112	101	2713.53	115.921
2015:04:09	18	12ID34	0	0	3599.93	0
2015:04:09	19	12ID34	0	0	3599.95	0
2015:04:09	20	12ID34	25	58	3599.95	658.324
2015:04:09	21	12ID34	132	9	3599.93	1114.98
2015:04:09	22	12ID34	86	46	3599.94	43.006
2015:04:09	23	12ID34	20	11	3599.94	64.091

Liite 7: Valokuva rakennetusta laitteesta